Universidade do Sul de Santa Catarina

Metodologias e Projetos de Software

Disciplina na modalidade a distância

Universidade do Sul de Santa Catarina

Metodologias e Projetos de Software

Disciplina na modalidade a distância

Palhoça
UnisulVirtual
2011

Créditos

Universidade do Sul de Santa Catarina | Campus UnisulVirtual | Educação Superior a Distância
Avenida dos Lagos, 41 – Cidade Universitária Pedra Branca | Palhoça – SC | 88137-900 | *Fone/fax*: (48) 3279-1242 e 3279-1271 | *E-mail*: cursovirtual@unisul.br | *Site*: www.unisul.br/unisulvirtual

Reitor
Ailton Nazareno Soares

Vice-Reitor
Sebastião Salésio Heerdt

Chefe de Gabinete da Reitoria
Willian Corrêa Máximo

Pró-Reitor de Ensino e Pró-Reitor de Pesquisa, Pós-Graduação e Inovação
Mauri Luiz Heerdt

Pró-Reitora de Administração Acadêmica
Miriam de Fátima Bora Rosa

Pró-Reitor de Desenvolvimento e Inovação Institucional
Valter Alves Schmitz Neto

Diretora do Campus Universitário de Tubarão
Milene Pacheco Kindermann

Diretor do Campus Universitário da Grande Florianópolis
Hércules Nunes de Araújo

Secretária-Geral de Ensino
Solange Antunes de Souza

Diretora do Campus Universitário UnisulVirtual
Jucimara Roesler

Equipe UnisulVirtual

Diretor Adjunto
Moacir Heerdt

Secretaria Executiva e Cerimonial
Jackson Schuelter Wiggers (Coord.)
Marcelo Fraiberg Machado
Tenille Catarina

Assessoria de Assuntos Internacionais
Murilo Matos Mendonça

Assessoria de Relação com Poder Público e Forças Armadas
Adenir Siqueira Viana
Walter Félix Cardoso Junior

Assessoria DAD - Disciplinas a Distância
Patrícia da Silva Meneghel (Coord.)
Carlos Alberto Areias
Cláudia Berh V. da Silva
Conceição Aparecida Kindermann
Luiz Fernando Meneghel
Renata Souza de A. Subtil

Assessoria de Inovação e Qualidade de EAD
Denia Falcão de Bittencourt (Coord.)
Andrea Ouriques Balbinot
Carmen Maria Cipriani Pandini

Assessoria de Tecnologia
Osmar de Oliveira Braz Júnior (Coord.)
Felipe Fernandes
Felipe Jacson de Freitas
Jefferson Amorin Oliveira
Phelipe Luiz Winter da Silva
Priscila da Silva
Rodrigo Battistotti Pimpão
Tamara Bruna Ferreira da Silva

Coordenação Cursos

Coordenadores de UNA
Diva Marília Flemming
Marciel Evangelista Catâneo
Roberto Iunskovski

Auxiliares de Coordenação
Ana Denise Goularte de Souza
Camile Martinelli Silveira
Fabiana Lange Patricio
Tânia Regina Goularte Waltemann

Coordenadores Graduação
Aloísio José Rodrigues
Ana Luísa Mülbert
Ana Paula R.Pacheco
Artur Beck Neto
Bernardino José da Silva
Charles Odair Cesconetto da Silva
Dilsa Mondardo
Diva Marília Flemming
Horácio Dutra Mello
Itamar Pedro Bevilaqua
Jairo Afonso Henkes
Janaína Baeta Neves
Jorge Alexandre Nogared Cardoso
José Carlos da Silva Junior
José Gabriel da Silva
José Humberto Dias de Toledo
Joseane Borges de Miranda
Luiz G. Buchmann Figueiredo
Marciel Evangelista Catâneo
Maria Cristina Schweitzer Veit
Maria da Graça Poyer
Mauro Faccioni Filho
Moacir Fogaça
Nélio Herzmann
Onei Tadeu Dutra
Patrícia Fontanella
Roberto Iunskovski
Rose Clér Estivalete Beche

Vice-Coordenadores Graduação
Adriana Santos Rammê
Bernardino José da Silva
Catia Melissa Silveira Rodrigues
Horácio Dutra Mello
Jardel Mendes Vieira
Joel Irineu Lohn
José Carlos Noronha de Oliveira
José Gabriel da Silva
José Humberto Dias de Toledo
Luciana Manfroi
Rogério Santos da Costa
Rosa Beatriz Madruga Pinheiro
Sergio Sell
Tatiana Lee Marques
Valnei Carlos Denardin
Sâmia Mônica Fortunato (Adjunta)

Coordenadores Pós-Graduação
Aloísio José Rodrigues
Anelise Leal Vieira Cubas
Bernardino José da Silva
Carmen Maria Cipriani Pandini
Daniela Ernani Monteiro Will
Giovani de Paula
Karla Leonora Dayse Nunes
Letícia Cristina Bizarro Barbosa
Luiz Otávio Botelho Lento
Roberto Iunskovski
Rodrigo Nunes Lunardelli
Rogério Santos da Costa
Thiago Coelho Soares
Vera Rejane Niedersberg Schuhmacher

Gerência Administração Acadêmica
Angelita Marçal Flores (Gerente)
Fernanda Farias

Secretaria de Ensino a Distância
Samara Josten Flores (Secretária de Ensino)
Giane dos Passos (Secretária Acadêmica)
Adenir Soares Júnior
Alessandro Alves da Silva
Andréa Luci Mandira
Cristina Mara Schauffert
Djeime Sammer Bortolotti
Douglas Silveira
Evilym Melo Livramento
Fabiano Silva Michels
Fabricio Botelho Espíndola
Felipe Wronski Henrique
Gisele Terezinha Cardoso Ferreira
Indyanara Ramos
Janaina Conceição
Jorge Luiz Vilhar Malaquias
Juliana Broering Martins
Luana Borges da Silva
Luana Tarsila Hellmann
Luíza Koing Zumblick
Maria José Rossetti
Marilene de Fátima Capeleto
Patricia A. Pereira de Carvalho
Paulo Lisboa Cordeiro
Paulo Mauricio Silveira Bubalo
Rosângela Mara Siegel
Simone Torres de Oliveira
Vanessa Pereira Santos Metzker
Vanilda Liordina Heerdt

Gestão Documental
Lamuniê Souza (Coord.)
Clair Maria Cardoso
Daniel Lucas de Medeiros
Jaliza Thizon de Bona
Guilherme Henrique Koerich
Josiane Leal
Marília Locks Fernandes

Gerência Administrativa e Financeira
Renato André Luz (Gerente)
Ana Luise Wehrle
Anderson Zandré Prudêncio
Daniel Contessa Lisboa
Naiara Jeremias da Rocha
Rafael Bourdot Back
Thais Helena Bonetti
Valmir Venício Inácio

Gerência de Ensino, Pesquisa e Extensão
Janaína Baeta Neves (Gerente)
Aracelli Araldi

Elaboração de Projeto
Carolina Hoeller da Silva Boing
Vanderlei Brasil
Francielle Arruda Rampelotte

Reconhecimento de Curso
Maria de Fátima Martins

Extensão
Maria Cristina Veit (Coord.)

Pesquisa
Daniela E. M. Will (Coord. PUIP, PUIC, PIBIC)
Mauro Faccioni Filho (Coord. Nuvem)

Pós-Graduação
Anelise Leal Vieira Cubas (Coord.)

Biblioteca
Salete Cecília e Souza (Coord.)
Paula Sanhudo da Silva
Marília Ignacio de Espíndola
Renan Felipe Cascaes

Gestão Docente e Discente
Enzo de Oliveira Moreira (Coord.)

Capacitação e Assessoria ao Docente
Alessandra de Oliveira (Assessoria)
Adriana Silveira
Alexandre Wagner da Rocha
Elaine Cristiane Surian (Capacitação)
Elizete De Marco
Fabiana Pereira
Iris de Souza Barros
Juliana Cardoso Esmeraldino
Maria Lina Moratelli Prado
Simone Zigunovas

Tutoria e Suporte
Anderson da Silveira (Núcleo Comunicação)
Claudia N. Nascimento (Núcleo Norte-Nordeste)
Maria Eugênia F. Celeghin (Núcleo Pólos)
Andreza Talles Cascais
Daniela Cassol Peres
Débora Cristina Silveira
Ednéia Araujo Alberto (Núcleo Sudeste)
Francine Cardoso da Silva
Janaina Conceição (Núcleo Sul)
Joice de Castro Peres
Karla F. Wisniewski Desengrini
Kelin Buss
Liana Ferreira
Luiz Antônio Pires
Maria Aparecida Teixeira
Mayara de Oliveira Bastos
Michael Mattar
Patrícia de Souza Amorim
Poliana Simao
Schenon Souza Preto

Gerência de Desenho e Desenvolvimento de Materiais Didáticos
Márcia Loch (Gerente)

Desenho Educacional
Cristina Klipp de Oliveira (Coord. Grad./DAD)
Roseli A. Rocha Moterle (Coord. Pós/Ext.)
Aline Cassol Daga
Aline Pimentel
Carmelita Schulze
Daniela Siqueira de Menezes
Delma Cristiane Morari
Eliete de Oliveira Costa
Eloísa Machado Seemann
Flavia Lumi Matuzawa
Geovania Japiassu Martins
Isabel Zoldan da Veiga Rambo
João Marcos de Souza Alves
Leandro Romanó Bamberg
Lygia Pereira
Lis Airê Fogolari
Luiz Henrique Milani Queriquelli
Marcelo Tavares de Souza Campos
Mariana Aparecida dos Santos
Marina Melhado Gomes da Silva
Marina Cabeda Egger Moellwald
Mirian Elizabet Hahmeyer Collares Elpo
Pâmella Rocha Flores da Silva
Rafael da Cunha Lara
Roberta de Fátima Martins
Roseli Aparecida Rocha Moterle
Sabrina Bleicher
Verônica Ribas Cúrcio

Acessibilidade
Vanessa de Andrade Manoel (Coord.)
Letícia Regiane Da Silva Tobal
Mariella Gloria Rodrigues
Vanesa Montagna

Avaliação da aprendizagem
Claudia Gabriela Dreher
Jaqueline Cardozo Polla
Nágila Cristina Hinckel
Sabrina Paula Soares Scaranto
Thayanny Aparecida B. da Conceição

Gerência de Logística
Jeferson Cassiano A. da Costa (Gerente)

Logísitca de Materiais
Carlos Eduardo D. da Silva (Coord.)
Abraao do Nascimento Germano
Bruna Maciel
Fernando Sardão da Silva
Fylippy Margino dos Santos
Guilherme Lentz
Marlon Eliseu Pereira
Pablo Varela da Silveira
Rubens Amorim
Yslann David Melo Cordeiro

Avaliações Presenciais
Graciele M. Lindenmayr (Coord.)
Ana Paula de Andrade
Angelica Cristina Gollo
Cristilaine Medeiros
Daiana Cristina Bortolotti
Delano Pinheiro Gomes
Edson Martins Rosa Junior
Fernando Steimbach
Fernando Oliveira Santos
Lisdeise Nunes Felipe
Marcelo Ramos
Marcio Ventura
Osni Jose Seidler Junior
Thais Bortolotti

Gerência de Marketing
Eliza B. Dallanhol Locks (Gerente)

Relacionamento com o Mercado
Alvaro José Souto

Relacionamento com Polos Presenciais
Alex Fabiano Wehrle (Coord.)
Jeferson Pandolfo
Karine Augusta Zanoni
Marcia Luz de Oliveira
Mayara Pereira Rosa
Luciana Tomadão Borguetti

Assuntos Jurídicos
Bruno Lucion Roso
Sheila Cristina Martins

Marketing Estratégico
Rafael Bavaresco Bongiolo

Portal e Comunicação
Catia Melissa Silveira Rodrigues
Andreia Drewes
Luiz Felipe Buchmann Figueiredo
Rafael Pessi

Gerência de Produção
Arthur Emmanuel F. Silveira (Gerente)
Francini Ferreira Dias

Design Visual
Pedro Paulo Alves Teixeira (Coord.)
Alberto Regis Elias
Alex Sandro Xavier
Anne Cristyne Pereira
Cristiano Neri Gonçalves Ribeiro
Daiana Ferreira Cassanego
Davi Pieper
Diogo Rafael da Silva
Edison Rodrigo Valim
Fernanda Fernandes
Frederico Trilha
Jordana Paula Schulka
Marcelo Neri da Silva
Nelson Rosa
Noemia Souza Mesquita
Oberdan Porto Leal Piantino

Multimídia
Sérgio Giron (Coord.)
Dandara Lemos Reynaldo
Cleber Magri
Fernando Gustav Soares Lima
Josué Lange

Conferência (e-OLA)
Carla Fabiana Feltrin Raimundo (Coord.)
Bruno Augusto Zunino
Gabriel Barbosa

Produção Industrial
Marcelo Bittencourt (Coord.)

Gerência Serviço de Atenção Integral ao Acadêmico
Maria Isabel Aragon (Gerente)
Ana Paula Batista Detóni
André Luiz Portes
Carolina Dias Damasceno
Cleide Inácio Goulart Seeman
Denise Fernandes
Francielle Fernandes
Holdrin Milet Brandão
Jenniffer Camargo
Jessica da Silva Bruchado
Jonatas Collaço de Souza
Juliana Cardoso da Silva
Juliana Elen Tizian
Kamilla Rosa
Mariana Souza
Marilene Fátima Capeleto
Maurício dos Santos Augusto
Maycon de Sousa Candido
Monique Napoli Ribeiro
Priscilla Geovana Pagani
Sabrina Mari Kawano Gonçalves
Scheila Cristina Martins
Taize Muller
Tatiane Crestani Trentin

Vera R. Niedersberg Schuhmacher

Metodologias e Projetos de Software

Livro didático

Design instrucional

Lívia da Cruz

6ª edição

Palhoça
UnisulVirtual
2011

Edição – Livro Didático

Professora Conteudista
Vera Rejane Niedersberg Schuhmacher

Design Instrucional
Dênia Falcão de Bittencourt
Viviane Bastos
Lívia da Cruz (5ª ed. rev. e atual.)

Assistente Acadêmico
Aline Cassol Daga (6ª edição)

ISBN
978-85-7817-291-6

Projeto Gráfico e Capa
Equipe UnisulVirtual

Diagramação
Jordana Paula Schulka (6ª edição)

Revisão
Fabricô

005.117
S41 Schuhmacher, Vera Rejane Niedersberg
Metodologias e projetos de software : livro didático / Vera Rejane Niedersberg Schuhmacher ; design instrucional Dênia Falcão de Bittencourt, Viviane Bastos, Lívia da Cruz ; [assistente acadêmico Aline Cassol Daga]. – 6. ed. – Palhoça : UnisulVirtual, 2011.
271 p. : il. ; 28 cm.

Inclui bibliografia.
ISBN 978-85-7817-291-6

1. Métodos orientado a objetos (Computação). 2. UML (Computação). 3. Software de aplicação. I. Bittencourt, Dênia Falcão de. II. Bastos, Viviane. III. Cruz, Lívia da. IV. Daga, Aline Cassol. V. Título.

Ficha catalográfica elaborada pela Biblioteca Universitária da Unisul

Sumário

Apresentação

Este livro didático corresponde à disciplina **Metodologias e Projetos de Software**.

O material foi elaborado visando a uma aprendizagem autônoma e aborda conteúdos especialmente selecionados e relacionados à sua área de formação. Ao adotar uma linguagem didática e dialógica, objetivamos facilitar seu estudo a distância, proporcionando condições favoráveis às múltiplas interações e a um aprendizado contextualizado e eficaz.

Lembre-se que sua caminhada, nesta disciplina, será acompanhada e monitorada constantemente pelo Sistema Tutorial da UnisulVirtual, por isso a "distância" fica caracterizada somente na modalidade de ensino que você optou para sua formação, pois na relação de aprendizagem professores e instituição estarão sempre conectados com você.

Então, sempre que sentir necessidade entre em contato; você tem à disposição diversas ferramentas e canais de acesso tais como: telefone, e-mail e o Espaço Unisul Virtual de Aprendizagem, que é o canal mais recomendado, pois tudo o que for enviado e recebido fica registrado para seu maior controle e comodidade. Nossa equipe técnica e pedagógica terá o maior prazer em lhe atender, pois sua aprendizagem é o nosso principal objetivo.

Bom estudo e sucesso!

Equipe UnisulVirtual.

Palavras da professora

Caro aluno/a,

A disciplina Metodologias e Projetos de Software vai inseri-lo/a no universo da modelagem de projetos de software.

O caminho da modelagem tem sido árduo através dos anos, sofrendo inúmeras inclusões e alterações, tudo para se adaptar às constantes evoluções da linguagem de programação, dos bancos de dados, dos sistemas operacionais, regras de negócio e paradigmas da programação.

A comunidade de desenvolvimento percebe a necessidade cada vez maior de documentar seus projetos, uma vez que o volume crescente de linhas de código torna nossa vida cada dia mais informatizada. Esse benefício, no entanto, cobra seu preço: os softwares sofrem manutenções constantes e as equipes são frequentemente modificadas, mas o arsenal de códigos continua tendo de ser executado com eficácia e eficiência.

Este é um dos contextos mais relevantes da modelagem: ela tem de permitir o entendimento do projeto a qualquer membro da equipe em qualquer ponto do processo, seja em uma etapa inicial de análise de requisitos ou já em fase de manutenção no cliente.

A modelagem pode ser comparada ao projeto de uma grande residência. Se você desejasse construir um pequeno canil e estivesse seguro/a sobre como fazê-lo, é bem provável que nem precisasse de um projeto para construí-lo, uma vez que a edificação seria pequena e não apresentaria muitos riscos. Agora, imagine a construção de uma casa de três andares. Será que você teria coragem de edificá-la sem a ajuda de um bom projeto que previsse todos os aspectos relevantes da obra?

Acho que você concorda comigo: nesse caso, nem pensaria em dispensar um projeto. Pois, nessa mesma relação, imagine a concepção de um software. Tenha a convicção de que a modelagem fiel do sistema proporcionará ganhos de qualidade e economia de recursos a longo prazo.

Então, convencido/a da importância do estudo desta disciplina? Pronto/a para começar?

Bons estudos!

Professora Vera Schuhmacher

Plano de estudo

O plano de estudos visa a orientá-lo no desenvolvimento da disciplina. Ele possui elementos que o ajudarão a conhecer o contexto da disciplina e a organizar o seu tempo de estudos.

O processo de ensino e aprendizagem na UnisulVirtual leva em conta instrumentos que se articulam e se complementam, portanto, a construção de competências se dá sobre a articulação de metodologias e por meio das diversas formas de ação/mediação.

São elementos desse processo:

- o livro didático;
- o Espaço UnisulVirtual de Aprendizagem (EVA);
- as atividades de avaliação (a distância, presenciais e de autoavaliação);
- o Sistema Tutorial.

Ementa

Análise de requisitos. Introdução ao *Rational Unified Process* (RUP). O paradigma orientado a objetos. Análise arquitetural. Modelagem de um sistema utilizando-se a notação UML: modelagem de use cases, análise e design; realização de use-case, diagrama geral de classes persistentes, diagrama de interfaces e mapeamento objeto-relacional.

Objetivos

Geral:

Elucidar ao aluno a importância da etapa de análise e modelagem do projeto de software e da necessidade de conhecimento de metodologias e notações que possam ser usados como facilitadores desta etapa.

Específicos:

- Propiciar ao/à aluno/a o conhecimento sobre conceitos relacionados ao ciclo de vida de desenvolvimento de um software.
- Sensibilizar o/a aluno/a sobre a importância do uso de metodologias de projeto de software para o sucesso efetivo deste projeto.
- Tornar presente a discussão sobre a relevância de se dispensar tempo suficiente para as etapas iniciais do projeto, como a análise de requisitos, procurando uma maturidade na compreensão das necessidades do usuário.
- Oferecer ao/à aluno/a o arsenal didático necessário para compreender o uso de técnicas e métodos estruturados de modelagem de sistemas.
- Capacitar o/a aluno/a a utilizar metodologias orientadas para a modelagem de sistemas.

Carga Horária

A carga horária total da disciplina é 120 horas-aula.

Conteúdo programático/objetivos

Veja, a seguir, as unidades que compõem o livro didático desta disciplina e os seus respectivos objetivos. Estes se referem aos resultados que você deverá alcançar ao final de uma etapa de estudo. Os objetivos de cada unidade definem o conjunto de conhecimentos que você deverá possuir para o desenvolvimento de habilidades e competências necessárias à sua formação.

Unidades de estudo: 9

Unidade 1 - ciclos de vida de um processo de desenvolvimento de software

Nesta unidade, são tratados aspectos relativos ao processo de desenvolvimento de software e seus possíveis modelos. Você terá a oportunidade de conhecer as etapas do processo de desenvolvimento de um software e perceberá que conduzir um processo de desenvolvimento de software exige um grande processo de gerência.

Unidade 2 - Engenharia de requisitos

Esta unidade aborda o reconhecimento da importância da etapa de engenharia de requisitos, técnicas e artefatos para o desenvolvimento do processo. Você vai perceber que a engenharia de requisitos é um processo interativo que envolve a compreensão do domínio, a coleta de requisitos, a estruturação e o estabelecimento de prioridades para a execução do projeto.

Unidade 3 - Análise estruturada

Esta unidade contempla a análise estruturada, em que são apresentados os requisitos da notação para uma abordagem estruturada. Notações e características específicas utilizadas em uma modelagem estruturada e que são fundamentais em sua documentação serão apresentadas.

Unidade 4 - Visão geral da UML

Nesta unidade, é feita a introdução à linguagem de notação UML, seus diagramas e as possíveis visões que iniciam a incursão na modelagem orientada a objetos, e são apresentadas as diferenças fundamentais existentes entre a análise estruturada e a análise orientada a objetos.

Unidade 5 - Modelagem de casos de uso

Esta unidade aborda o diagrama de casos de uso, sua documentação, nomenclatura e a identificação dos atores. Você terá a oportunidade de identificar e construir os casos de uso necessários para descrever o projeto.

Unidade 6 - Modelagem de classes

Nesta unidade, você perceberá que o modelo estático da UML apresenta o diagrama de classes, responsabilidades, relacionamentos e a divisão das classes do modelo de análise. O modelo de classes é um dos modelos mais ricos em termos de notação e concentra o cerne estático de todo o projeto.

Unidade 7 - Modelos de interações

Nesta unidade, você vai estudar sobre o modelo de interações, que apresenta as mensagens trocadas entre os objetos na execução de um determinado cenário. O uso do modelo de interações procura descrever o modelo dinâmico do sistema propondo a descrição da troca de mensagens entre objetos.

Unidade 8 - Modelos de estados

O modelo de estados e atividades é apresentado nesta unidade. Você conhecerá os diagramas de transição de estado que modelam o comportamento de um objeto e o diagrama de atividade que modela a sequência geral de ações para vários objetos e casos de uso.

Unidade 9 - RUP e ICONIX

Nesta unidade do livro você será apresentado aos modelos RUP e ICONIX e estudará sobre seus conceitos, fases e elementos. Você perceberá como o RUP colabora para a estruturação efetiva de tarefas e fluxos de trabalho de profissionais, atuando em equipes de desenvolvimento de software.

Agenda de atividades/Cronograma

- Verifique com atenção o EVA, organize-se para acessar periodicamente a sala da disciplina. O sucesso nos seus estudos depende da priorização do tempo para a leitura, da realização de análises e sínteses do conteúdo e da interação com os seus colegas e professor.
- Não perca os prazos das atividades. Registre no espaço a seguir as datas com base no cronograma da disciplina disponibilizado no EVA.
- Use o quadro para agendar e programar as atividades relativas ao desenvolvimento da disciplina.

Atividades obrigatórias	
Demais atividades (registro pessoal)	

UNIDADE 1

Ciclos de vida de um processo de desenvolvimento de software

Objetivos de aprendizagem

- Compreender as características do produto de software.
- Perceber as diversas etapas do processo de desenvolvimento de um software.
- Definir essas etapas dentro de um modelo de desenvolvimento de projeto.

Seções de estudo

Para início de estudo

Desenvolver software*s* é uma tarefa árdua e complexa. Lidar com essa complexidade significa compreender claramente os processos relacionados ao desenvolvimento do software.

É preciso entender claramente o que cobrar e como deve ser cobrado em cada processo, o que pode ser executado em paralelo ou de forma sequencial. Esse conjunto de atribuições constitui as atividades de gerência. Conduzir o processo de desenvolvimento de software é um grande processo de gerência e, para guiá-lo da melhor forma possível, é preciso saber o que se espera de cada etapa.

Nesta unidade, você vai perceber que todo o processo de desenvolvimento, por mais complexo que pareça, passa por etapas pré-definidas e universais. Essas etapas são conduzidas de forma diferente, dependendo da natureza do projeto ou mesmo da cultura de cada empresa. Definir a forma como esses processos serão executados é fundamental para o bom andamento de todo o processo e até mesmo para a escolha de metodologias futuras.

Vamos iniciar esta jornada metodológica? Bom estudo!

Seção 1 – Quais são as características do software?

Termos como internet, *web* e software são modernos e estão na mídia diariamente. São termos que, antigamente, eram exclusivos de áreas extremamente técnicas e agora se tornaram de domínio público.

Ainda que informalmente se tenha uma ideia básica do significado de internet ou de software, é importante que se conheça o seu verdadeiro significado. Sendo assim, você sabe o que significa software?

Será que você sabe realmente o que significa software?

Explicar esse conceito não é simples, depende da ótica pela qual se tenta entender.

Você poderia dizer que software é o conjunto de instruções que, ao serem executadas, produzem a função e o desempenho desejados.

Poderia conceituar software como estruturas de dados que possibilitam que os programas manipulem adequadamente a informação.

É possível dizer, ainda, que softwares são os documentos que descrevem a operação e o uso dos programas desenvolvidos ou projetados por engenharia, não manufaturados no sentido clássico.

Todavia, é possível ser redundante e dizer que softwares são ferramentas pelas quais se explora os recursos do *hardware*, executa-se determinadas tarefas, resolve-se problemas ao interagir com a máquina e tornam o computador operacional.

Conceituar software passa por conhecer suas características. Vejamos:

O software apresenta características únicas. Ele não se desgasta, mas se deteriora.

Mas o que é isso?

Imagine que você compre uma Ferrari novinha!!! Agora, imagine você circulando com a Ferrari todos os dias a 200 km/h por uma estrada cheia de buracos.

Como estará sua Ferrari daqui a um ano?

Com certeza vai estar cheia de ruídos e talvez apresente problemas em alguns componentes. Isso acontece devido ao

desgaste do carro. Ao final de dez anos dirigindo o carro por essa mesma estrada, é bem provável que você tenha um carro cheio de problemas!

Agora, imagine que você tenha uma videolocadora e resolva comprar um software para informatizar todo o processo de atendimento. Abstraindo questões relacionadas ao *hardware*, esse software daqui a 100 anos irá funcionar da mesma maneira que hoje, ou seja, se tiver algum problema de programação, repetirá o mesmo problema em todas as vezes que for executado. Mas se o software não tiver problemas, rodará perfeitamente todas as vezes que você solicitar sua execução.

Ao final de três anos, no entanto, mesmo funcionando perfeitamente, talvez esse software não supra mais suas necessidades. Imagine que nesse sistema você digite o código do DVD para realizar a movimentação, mas agora gostaria que ele lesse o código de barras do DVD. E agora?

É por isso que dizemos que o software não se desgasta, mas se deteriora em função de inovações tecnológicas e de novas necessidades do cliente. Ele acaba não atendendo mais às suas necessidades, apesar de funcionar, muitas vezes, sem apresentar.

O *software* é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico.

Quando você comprou sua geladeira, em algum momento pensou que ela foi fabricada como uma peça única? Claro que não! Ela foi produzida em uma linha de produção.

Agora pense em uma linha de produção de caminhões. Os componentes são incorporados ao projeto para dezenas de unidades.

Mas quando se pensa em software, sabe-se que cada produto é projetado e pensado como uma peça única. É impossível visualizá-lo em uma esteira de produção!

Imagine as seguintes situações:

- Na linha de produção de caminhões, saem da fábrica 30 unidades diárias, 600 por mês. Dessas 600 unidades, a indústria possui uma estatística de problemas apresentados em 0,2% da produção durante sua utilização pelo cliente nos primeiros seis meses. Isso significa uma margem de problemas em um caminhão saído da linha de produção mensal. Essa informação pode acabar com o nome da empresa? Qual sua opinião?
- Agora imagine uma empresa de software. Ela desenvolve o famoso sistema de videolocadora e vende esse produto para 600 clientes. Mas no final do sexto mês, acontece um erro no sistema (por exemplo, o programa não está armazenando corretamente a data de entrega do DVD). Essa mesma versão foi entregue para os 600 clientes e teremos então 600 clientes com o mesmo problema. E agora? Essa informação pode acabar com o nome da empresa? Qual sua opinião?
- **Sintetizando o problema:** Em uma fábrica de software, o sucesso é medido pela qualidade de uma única entidade e não pela qualidade de muitas entidades manufaturadas.

Seção 2 – Quais são as etapas do processo de desenvolvimento de software?

Existem alguns conceitos que são fundamentais quando falamos de desenvolvimento, e o primeiro deles é **Engenharia de Software (EGS)**. Assim como a palavra software, Engenharia de *Software* também é um termo que gera discussões sobre a melhor maneira de expor todas as suas nuances.

A **IEEE** definiu Engenharia de *Software* como a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, a operação e a manutenção do software; o estudo de abordagens e princípios, a fim de se obterem economicamente softwares confiáveis e que sejam executados de forma eficiente em máquinas reais.

IEEE - Instituto de Engenharia Elétrica e Eletrônica (IEEE) é formado pela fusão do Instituto de Engenheiros de Rádio (IRA) com o Instituto Americano de Engenheiros Elétricos (AIEE). A meta do IEEE é promover conhecimento no campo da engenharia elétrica e eletrônica estabelecendo padrões para formatos de computadores e dispositivos.

A primeira definição do termo foi dada por Fritz Bauer em 1969:

> Engenharia de software é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais.

A partir desse conceito, é importante que você entenda o significado de método, procedimento e ferramenta, que são fonte de muita confusão, segundo Pressman (2002):

- **Métodos** - proporcionam os detalhes de "como fazer" para construir o *software*. Os métodos envolvem um amplo conjunto de tarefas, como o planejamento e estimativa do projeto, a análise de requisitos de *software*, o projeto da estrutura de dados, o algoritmo de processamento, a codificação, o teste e a manutenção.
- **Ferramentas** - fornecem suporte automatizado aos métodos. Atualmente existem ferramentas para sustentar cada um dos métodos. Quando as ferramentas são integradas, é estabelecido um sistema de suporte ao desenvolvimento de *software*, chamado *Computer Aided Software Engineering (CASE*).
- **Procedimentos** - constituem o elo entre os métodos e ferramentas, eles estabelecem a sequência em que os métodos serão aplicados, os produtos (*deliverables*) que devem ser entregues, os controles que ajudam a assegurar a qualidade e coordenar as alterações e os marcos de referência que possibilitam administrar o progresso do *software*.

Quando se fala em produção de *software*, é muito difícil estabelecer preço, prazo e número de pessoas necessárias para o desenvolvimento. Existe uma metodologia chamada **ponto por função** que nos ensina a encontrar resultados para as dúvidas relacionadas a essas estimativas. Como fazer isso manualmente é muito difícil, foram desenvolvidas ferramentas baseadas nesse método, que o automatizam, mas, **o ponto por função é uma metodologia que estabelece entrada de informações e resultados na forma de cálculos e relatórios**. Os procedimentos serão utilizados para dizer quando as entradas de informações serão feitas, quais relatórios serão emitidos e em que momento.

Segundo **Pádua** (2001), a EGS trata do software como produto e, como todo produto industrial, o software:

Wilson de Pádua Paula Filho será referenciado como Pádua neste livro pelo fato de ser assim conhecido na área

- deve ser concebido a partir da percepção de uma necessidade;
- é desenvolvido transformando-se em um conjunto de itens entregue ao cliente;
- entra em operação utilizado dentro de um processo de negócio e sujeito à manutenção quando necessário;
- é retirado de operação ao final de sua vida útil.

O que é processo de *software*?

Quando se fala em processo longo, nos vem à mente uma sequência de passos para atingir um objetivo. E, como afirma Pádua (2001), processo de software é exatamente isto: um conjunto de passos ordenados, constituídos por atividades, métodos, práticas e transformações, usado para atingir uma meta.

Como você já deve estar imaginando, processos podem ser definidos para atividades em todas as etapas do desenvolvimento de um software. Pressman (2002) sugere a divisão do desenvolvimento em três processos genéricos. Acompanhe a figura:

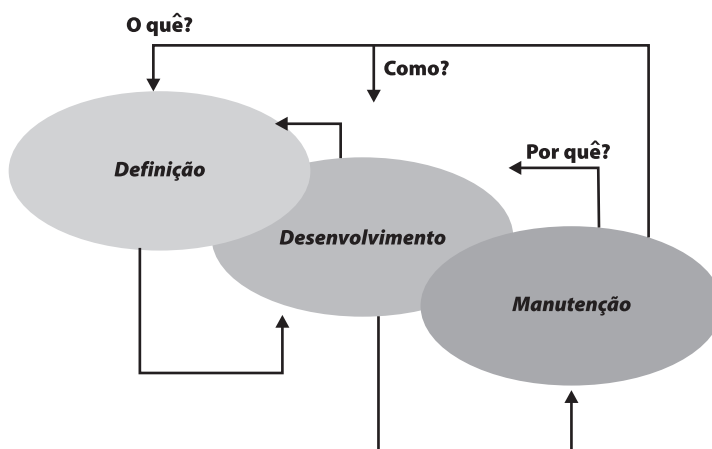



Figura 1.1 – Etapas do desenvolvimento de software
Fonte: Pressman (2002).

Cada etapa pode ser detalhada como:

a) Definição

A primeira etapa constitui-se por identificar quais informações devem ser processadas, qual função e desempenho são desejados, quais interfaces devem ser estabelecidas, quais restrições de projeto existem (por exemplo, o cliente precisa do software em 60 dias!) e quais critérios de avaliação são exigidos para se definir um sistema bem-sucedido (existem, por exemplo, normas internacionais que devem ser obedecidas no chão da fábrica e ser incorporadas ao sistema).

A etapa de definição subdivide-se em:

- **Planejamento do projeto** – no planejamento avaliamos se o projeto é viável, e como o desenvolvimento será conduzido.

- **Levantamento de requisitos** – Nessa etapa, o objetivo principal é compreender o problema do cliente, suas necessidades, expectativas e restrições. Aqui você deve pensar com o cliente. Quais problemas existem? Como eles acontecem? Existe alguma característica específica que deve ser observada? De velocidade, portabilidade, interface, por exemplo.

- **Análise de requisitos** – Etapa de análise dos dados da etapa anterior. Na análise de requisitos ou especificação de requisitos, concebe-se uma estratégia para solucionar problemas e necessidades do cliente. Nessa etapa, você ainda não precisa se preocupar com questões tecnológicas. Assim, você vai dizer o que o sistema deve fazer para apoiar o cliente, a partir de modelos.

b) Desenvolvimento

Nesta fase, você tenta definir como a estrutura de dados e a arquitetura do software têm de ser projetadas. Durante a segunda etapa, você irá realizar:

- **O projeto de *software*** – no projeto de *software*, você deve definir como o sistema vai funcionar para atender aos requisitos levantados. Nessa fase, temos como resultado uma descrição computacional daquilo que o *software* deve fazer. Normalmente, tem-se aqui o projeto da arquitetura (especificação da configuração de componentes do *software* – funções, classes, objetos e suas interconexões) e o projeto detalhado (que tem por objetivo a concepção e a especificação das estruturas de dados e dos algoritmos, que realizam aquilo que foi especificado para cada componente do *software*).

- **Implementação** – Na implementação, o sistema é codificado.

- **Teste de *software*** – Na etapa de testes ocorre a verificação do que foi implementado. Várias são as técnicas: desde corretivas (procura de erros) até de validação com grupos de usuários.

c) Manutenção

Nessa fase ocorrerão mudanças no software em consequência dos erros encontrados. Além disso, é a etapa responsável pelas adaptações do software em função da evolução do hardware e necessidades do cliente.

A manutenção pode ser corretiva (o cliente encontrará defeitos no software), adaptativa (modificações no software que acomodam mudanças relacionadas à evolução do hardware: por exemplo, o cliente decide trocar o sistema operacional por uma possibilidade *open source*) e funcional (implementa funções adicionais para o cliente relacionadas à expectativa que ele não tinha durante o desenvolvimento, como em uma situação na qual o gerente de recursos humanos precisa de um novo relatório estatístico).

Além das três etapas básicas, devem ser consideradas ainda as atividades complementares fundamentais para o bom andamento do processo:

- **Revisões** – Efetuadas para garantir que a qualidade seja mantida à medida que cada etapa é concluída.
- **Documentação** – É desenvolvida e controlada para garantir que informações completas sobre o *software* estejam disponíveis para uso posterior.
- **Controle das mudanças** – É instituído de forma que as mudanças possam ser aprovadas e acompanhadas pelos gerentes e pela equipe de projeto.

Seção 3 – Modelo de desenvolvimento de *software*

O processo de desenvolvimento apresentado na seção 2 é um modelo genérico. Isso significa que suas etapas são aplicáveis a qualquer modelo, independentemente do paradigma de desenvolvimento utilizado. As diferentes formas de proceder em relação ao processo de desenvolvimento apresentam características próprias a cada modelo, diferentes necessidades das empresas desenvolvedoras de software ou mesmo ao tipo de sistema desenvolvido.

Pressman (2002) define o modelo de desenvolvimento como uma representação abstrata do processo de desenvolvimento que define como as etapas relativas ao desenvolvimento de software serão conduzidas e inter-relacionadas para atingir o objetivo do desenvolvimento do software.

Alguns modelos de desenvolvimento estão há muitos anos no mercado, outros são muito recentes. Dê uma olhadinha, a seguir, nos mais conhecidos por sua utilização nas empresas.

a) Modelo cascata

No modelo cascata, os subprocessos são executados em uma sequência rígida. Assim, cada subprocesso passa a ser um marco de controle. Esse modelo exige uma abordagem sistemática, sequencial, no desenvolvimento de software.

Isso acontece da seguinte forma, o desenvolvedor visita a empresa e inicia a etapa de levantamento de requisitos, após algumas sessões, finaliza essa etapa e inicia outra, denominada etapa de análise.

Finalizada a etapa de análise, o processo de desenho do sistema, de todo seu projeto. Finalizada esta etapa inicia-se a implantação do código.

Observe que não existe retorno a nenhuma etapa anterior. O modelo é inflexível: as etapas são sequenciais e não permitem regresso. Nesse modelo, ao finalizar o desenho do projeto, o desenvolvedor só poderá modificá-lo na etapa de manutenção.

Essa visão burocrática dos subprocessos gera situações difíceis de resolver pelos analistas. Pense: projetos reais raramente seguem o fluxo sequencial que o modelo propõe. No início de um projeto, é difícil estabelecer explicitamente todos os requisitos, existe uma incerteza natural. Mas, o maior problema de todos é a falta de visibilidade para o cliente! Somente na etapa de implantação o cliente terá contato com o software.

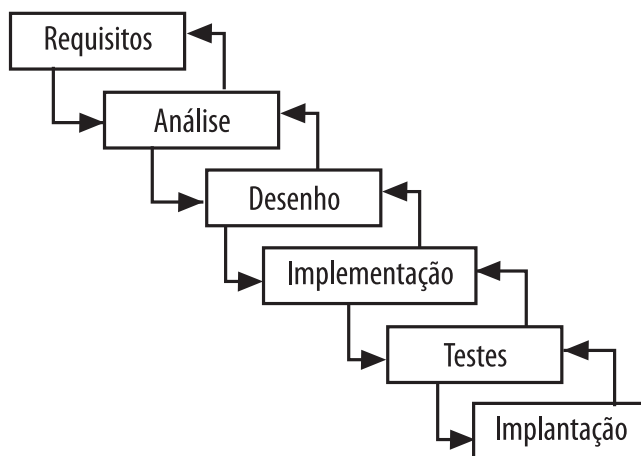



Figura 1.2 – Modelo cascata
Fonte: Pádua (2001).

No modelo em cascata, todo o projeto deve ser entendido em suas fases iniciais, seguindo até o final com pouquíssima interação com o cliente.

Imagine a situação: você é contratado para desenvolver um projeto em uma fábrica de calçados na qual todo o processo de produção será automatizado. Em seu planejamento, percebe que o processo todo estará finalizado em um prazo de 2 anos. Se você optar pelo modelo cascata, o cliente será apresentado ao sistema somente ao final de 2 anos! Será que ele terá paciência para esperar por este prazotodo esse tempo?

Outro ponto importante: o levantamento de requisitos é feito no início e o processo segue para a etapa seguinte sem retornar à anterior. Será que, em dois anos, nada vai mudar no processo da empresa?

Será, então, que este modelo é inviável para os tempos atuais? Saiba que foi um dos modelos mais utilizados no mundo e tem baixo custo, devido a sua estrutura sequencial. Nos tempos atuais podemos sugeri-lo quando se tem domínio do problema e/ou o projeto é considerado pequeno.

b) Modelo espiral

O modelo em espiral é totalmente diferente do anterior. Nele, a palavra de ordem é a experimentação e a avaliação.

Esse modelo é desenvolvido em uma série de interações; cada interação corresponde a uma volta na espiral. Cada volta é fundamentada na análise de riscos da solução que está sendo proposta.

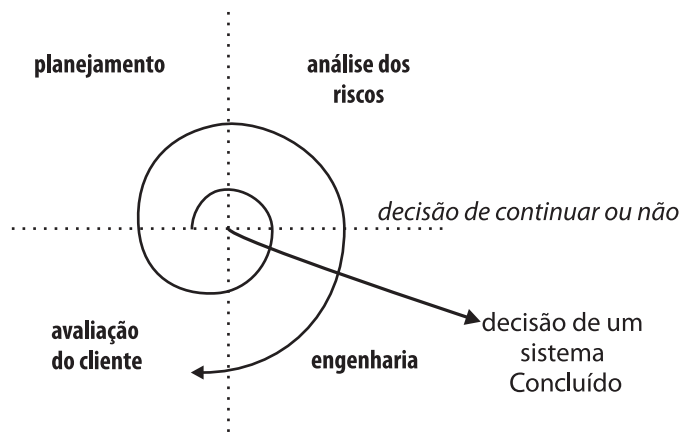



Figura 1.3 – Modelo espiral
Fonte: Pressman (2002).

Na etapa de planejamento são determinados os objetivos, alternativas e restrições. Durante o subprocesso de análise de risco, são analisadas as alternativas e identificados os riscos e resoluções possíveis. Na construção ocorre o desenvolvimento do produto no nível seguinte. A avaliação do cliente é fundamental, pois nela ocorre a avaliação do produto e o planejamento das novas fases – cliente e desenvolvedor refinam os requisitos dos software*s* a serem desenvolvidos.

O modelo exige dos desenvolvedores experiência na determinação de riscos, sendo portanto, dependente da experiência pessoal da equipe.

Talvez você esteja se perguntando quando poderá utilizar esse modelo. É uma boa pergunta. Imagine que você tenha de apresentar uma solução para um cliente e que você não esteja certo sobre a tecnologia a ser utilizada ou a forma como a estrutura de dados deva ser construída. A melhor maneira de fazer o "melhor serviço" será usando o modelo espiral.

Utilize prototipação, simulação ou qualquer recurso possível para avaliar diferentes soluções até encontrar a melhor solução! Quando você estiver certo da resposta, siga na direção da engenharia, onde temos as etapas padrão do processo de desenvolvimento.

Imagine que você foi contratado para desenvolver um produto que faça vídeoinspeção a distância de dutos de galerias de drenagem para a prefeitura do Rio de Janeiro. Você poderia indicar uma solução de imediato? Ou teria dúvidas sobre como fazê-lo da melhor maneira, pela grande oferta de recursos tecnológicos e mesmo restrições de projeto existentes?

c) Modelo prototipação

A prototipação envolve a produção de versões iniciais – "protótipos" – de um sistema futuro com o qual podem-se realizar verificações e experimentações para avaliação de algumas de suas qualidades antes que o sistema venha realmente a ser construído.

Esse modelo é popularmente usado como um mecanismo para identificar os requisitos de software.

Imagine você iniciando um trabalho com um cliente que lhe solicita um software para o controle de uma UTI, mas você não faz ideia do funcionamento de uma UTI. E o pior é que o cliente não parece dispor de muito tempo para explicar novamente o que você não conseguiu entender.

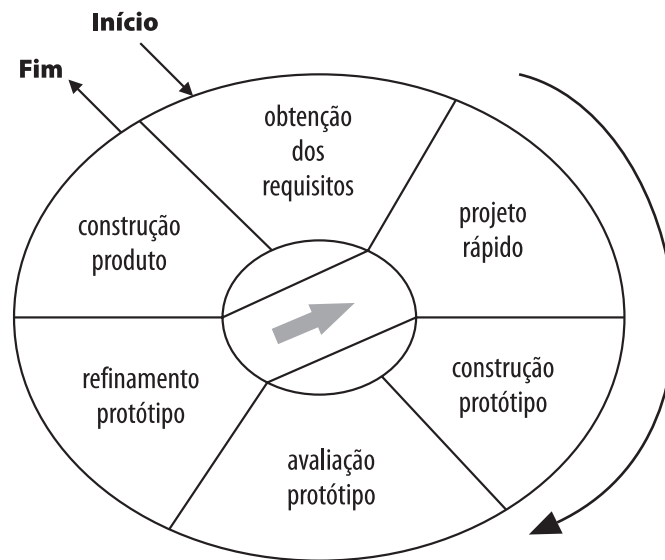



Figura 1.4 – Modelo prototipação
Fonte: Pressman (2002).

Na figura 1.4 você percebe o ciclo de desenvolvimento da prototipação: o processo inicia com a obtenção de requisitos, segue-se então um projeto rápido para o futuro desenvolvimento do protótipo. Finalizado o protótipo, é realizada uma avaliação com o cliente ou com a própria equipe de projeto. Os resultados da avaliação são utilizados para refinar o protótipo, reiniciando o ciclo até a avaliação.

O processo se repete até o momento em que o grau de aceitação do protótipo seja considerado aceitável. A partir deste momento, inicia-se a construção do produto, em que se pode utilizar qualquer outro modelo: cascata, incremental, entre outros.

A prototipação pode ser conduzida segundo duas técnicas específicas de construção:

- a *prototipação horizontal*, em que uma camada específica do sistema é construída (a interface do usuário com suas janelas e *widgets* ou camadas da aplicação, como as funções para transação em bancos de dados);

- a *prototipação vertical*, em que uma parte da funcionalidade do sistema é escolhida para ser implementada completamente. A prototipação vertical é interessante quando aspectos da funcionalidade não estão claros. A partir da validação do protótipo, o modelo segue no processo tradicional de desenvolvimento para a construção do produto.

Lembre-se: quando se valida uma informação verbal com o cliente, em muitos casos a atenção dele não é total e muitas palavras se perdem durante o diálogo. Um desenho, no papel, do protótipo de uma tela torna-se 100 vezes mais claro e objetivo, além de conseguirmos 100% da atenção do cliente.

d) Modelo incremental

O modelo incremental foi desenvolvido a partir da combinação entre os modelos linear e de prototipação (PRESSMAN, 2002). Quando você usa esse modelo, todo o desenvolvimento é dividido em etapas que são produzidas de forma incremental até se chegar a um sistema finalizado.

Para cada etapa é realizado um ciclo completo de desenvolvimento e novas funcionalidades são adicionadas ao sistema. Assim, você pode dizer que todo o desenvolvimento evolui em versões.

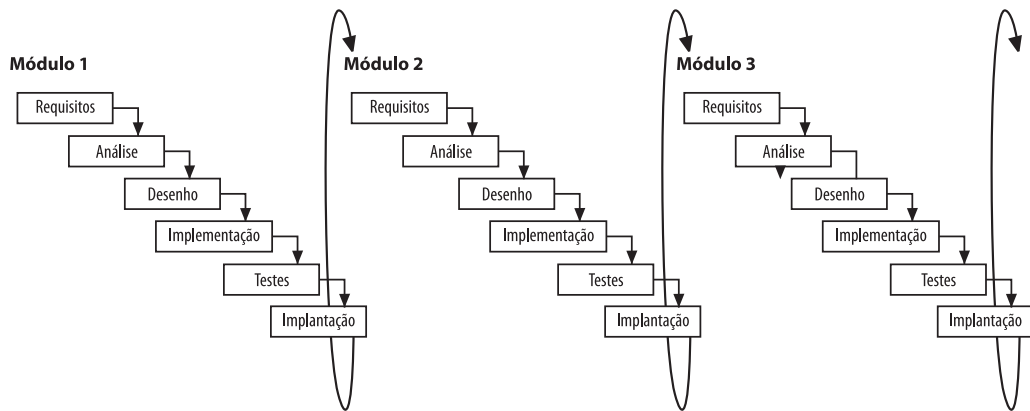



Figura 1.5 – Modelo incremental
Fonte: Pádua (2001).

Se você deseja utilizar esse modelo, considere a forma como todo o projeto será construído. Pense em questões relacionadas à prioridade (o que é mais importante para o cliente) e no risco de cada requisito. Tenha estratégia em mente e observe se o que é prioritário pode ser desenvolvido primeiro. Não esqueça: o primeiro módulo é o cartão de visitas de sua empresa para o cliente!

Nesse modelo, temos uma forte participação do cliente no projeto. Ele avalia os incrementos emtregues, o que permite corrigir possíveis problemas nos módulos em desenvolvimento e realizar os acertos devidos. Outro ponto importante é que você deve considerar, logo no início, os requisitos mais arriscados. Fazendo isso, qualquer inconsistência aparece logo no começo e você terá tempo para reagir e solucionar o problema.

Fundamental nesse modelo é a visão global do sistema. O gerente deve ter em mente a soma de todos os módulos, evitando redundâncias na construção do produto.

O modelo incremental é um dos mais utilizados na atualidade e em sua estrutura é comum percebermos a inserção da prototipação.

E as metodologias ágeis?

Até pouco tempo, tudo o que se pensava sobre metodologias de desenvolvimento de software parecia extremamente ligado a farta documentação e processos delineados com clareza. Para muitos desenvolvedores, no entanto, passos firmemente delineados e documentados para cada etapa do desenvolvimento eram considerados uma burocracia desnecessária e que por vezes produzia altos custos ao projeto. A divergência sobre a eficiência do uso de modelos que consideravam aqueles mais tradicionais, limitadores para equipes de desenvolvimento e, não raro, eram apontados como causa de atrasos e dificuldades no desenvolvimento. Por outro lado, pequenas empresas disprovidas de grandes orçamentos ficavam à margem dos modelos tradicionais por sua incapacidade de comportar a estrutura necessária para sua execução.

A demanda criada a partir dessa situação deu origem a um novo modelo, segundo Soares (2004), voltado a pessoas e não a processos ou algoritmos: os métodos ágeis. As metodologias ágeis possuem características que determinam um foco maior na codificação e não na documentação, são claramente adaptativas, pois novos fatos que são apresentados no decorrer do projeto são tratados durante o desenvolvimento, e não existe a preocupação de conhecer o todo a partir de uma avaliação preditiva: novidades são adaptadas e incorporadas durante o desenvolvimento. A metodologia ágil é, portanto, naturalmente interativa, incremental, e acrescenta uma boa participação do usuário no processo.

A seguir, conheça duas representantes dessa nova corrente, as metodologias *Extreme Programming* e SCRUM.

Extreme Programming (XP)

O método XP foi criado por Kent Beck na década de 90, e no ano de 1996 tornou-se mundialmente conhecido ao ser utilizado na empresa Daimler Chrysler. Havia um projeto na empresa que durante anos não fora concluído e, ao fazer uso da metodologia, teve sua execução completa realizada em apenas quatro meses.

Mas o que é o XP? Segundo Beck (1999), o XP é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente.

Nesse processo, a burocracia deve ser mínima, as equipes, pequenas (sugere-se até 10 pessoas), trabalhando de forma incremental, apoiados integralmente pelo futuro usuário do sistema. A análise dos requisitos ocorre à medida que são descritos e solicitados pelo usuário, ou seja, o conhecimento sobre o problema evolui durante seu desenvolvimento.

O XP é norteado por valores, princípios e regras. Beck (1999) define quatro valores que descrevem o XP: a comunicação, a simplicidade, o *feedback* e uma boa dose de coragem. A seguir, acompanhe uma breve descrição desses valores, segundo Wuestefeld (2001).

- **Simplicidade** – Está ligada à simplicidade do software, do processo usado no desenvolvimento, na comunicação e em tudo o que norteia o trabalho de desenvolvimento, ou seja, descomplicar. A regra do XP é simplificar!

- **Comunicação** – Ela deve ocorrer em todos os sentidos, desde o uso dos escritos a partir da convenção de padrões de codificação para facilitar o entendimento até a preferência por reuniões (presenciais) para discussão de requisitos e dúvidas, o trabalho em salas comuns evitando-se o isolamento da equipe, a execução de revisões do projeto em conjunto com toda a equipe.

- ***Feedback*** – Os possíveis problemas devem ser identificados o mais cedo possível para que possam ser corrigidos rapidamente. Da mesma forma, oportunidades descobertas devem ser aproveitadas rapidamente.

- **Coragem** – É fundamental para apontar problemas, solicitar ajuda, para alterar e simplificar um que já esteja funcionando, para apresentar prazos que muitas vezes podem não agradar ao seu usuário final, mas que são necessários para não comprometer a qualidade do projeto.

Os valores do XP são fundamentados em alguns princípios como o *feedback* rápido, a simplicidade que deve ser assumida em todas as etapas do projeto, a consciência de mudanças que devem acontecer de forma incremental, a compreensão e aceitação das mudanças e a necessidade da qualidade do trabalho.

Valores e princípios se entrelaçam na definição de práticas básicas que são adotadas pelo XP (BONA, 2002).

As doze práticas básicas adotadas pelo XP segundo Soares (2004) são:

- **Jogo de planejamento**. Nesta prática, decide-se o que deve ser feito (requisitos) e as prioridades. Também é determinado o escopo da próxima versão. Divide-se por duas áreas: o jogo do planejamento que decide sobre o escopo, a composição das versões e as datas de entrega é realizado pela área de negócios; as estimativas de prazo, o processo de desenvolvimento e o cronograma são determinados pelos programadores.

- **Programação em pares**. A programação é feita em duplas, dois programadores trabalham sempre juntos na mesma máquina. Enquanto um dos programadores escreve, o segundo confere a padronização e o raciocínio lógico.

- **Semana de 40 horas**. No modelo, a semana é de 40 (quarenta) horas. Hora extra é abolida, pois parte-se do princípio de que um programador cansado produz mais erros e, portanto, deve ser evitado.

- **Pequenas versões**. O sistema é baseado em entregas frequentes de versões com tamanho pequeno, que são incrementadas em requisitos continuamente (entregas de versões mensais com *feedback* de aceitação do cliente, por exemplo).

- **Metáforas**. O uso de metáforas procura descrever o software sem a utilização de termos técnicos, facilitando a comunicação com o cliente.

- **Projeto simples**. O sistema precisa ser projetado o mais simplesmente possível, satisfazendo os requisitos atuais, sem preocupação com requisitos futuros.

- **Teste**. A equipe de desenvolvimento descreve em um primeiro momento os casos de testes, após essa tarefa continua o desenvolvimento. Assim como os desenvolvedores, os clientes também escrevem testes a fim de validar o atendimento dos requisitos.

- ***Refactoring***. Sempre que possível, o projeto deve ser aperfeiçoado. Esse aperfeiçoamento deve propor a clareza do software.

- **Propriedade coletiva**. Pertence a todo o grupo. Isso significa que não existe o conceito de propriedade para um algoritmo. Qualquer integrante da equipe pode acrescentar ou alterar, desde que realize os testes necessários para validá-lo.

- **Integração contínua**. A construção e a integração do sistema ocorrem várias vezes por dia. Tarefas são completadas continuamente.

- **Cliente dedicado**. O usuário final está disponível todo o tempo, determinando os requisitos, atribuindo prioridades e, principalmente, respondendo às dúvidas.

- **Padronização do código**. O código é escrito com um grande cuidado quanto a padronizações. Isso facilita o entendimento, assegurando a clareza e a comunicação entre programadores e projetistas .

SCRUM (*Software Development Process*)

Segundo relatos de Sutherland (2004), o SCRUM foi documentado e concebido na empresa *Easel Corporation* em 1993. A ideia inicial era voltada para empresas automotivas. Mais tarde, a metodologia foi incorporada pelas empresas *Advanced Development Methods* e VMARK, passando por modificações e melhorias até chegar em um processo considerado adequado para projetos e desenvolvimento de software orientado a objetos.

O SCRUM possui aspectos semelhantes ao XP: também é formatado para o uso de equipes pequenas, nas quais os requisitos não são claros, com interações de curta duração e o máximo de visibilidade no processo de desenvolvimento.

O SCRUM divide o desenvolvimento em interações com duração de 30 dias, chamadas de *sprints*. As equipes multidisciplinares são formadas por projetistas, programadores, engenheiros e gerentes de qualidade. Cada equipe trabalha em torno dos requisitos do projeto que são definidos no início de um *sprint*. A cada dia ocorre uma reunião da equipe, na qual se discutem metas, dificuldades e objetivos. Observe que isso acontece *todos os dias*!

Quais são as características da metodologia SCRUM?

Schwaber (2002) algumas características da metodologia SCRUM que caracterizam etapas do seu processo de desenvolvimento.

Existem duas fases bem definidas: a primeira é de planejamento (pré-*game phase*) e a segunda, de fechamento (*post-game phase*), onde os processos são definidos claramente.

Durante o planejamento, definem-se requisitos, são estabelecidas prioridades e estimativas e são feitas análises de necessidades, como treinamento da equipe e sobre as ferramentas. Na fase de *sprint* (*game phase*), todo o processo baseia-se na experiência da equipe. As fases do *sprint* são gerenciadas por meio de controles que abrangem riscos e a otimização de questões de flexibilidade. Os *sprints* não apresentam uma estrutura linear: muitas vezes, o uso da tentativa e do erro funciona para determinar e reconhecer ou conhecer o processo. Ainda assim, é importante entender que o *sprint* acontece de forma tradicional, ou seja, seguindo etapas de análise, projeto, implementação e testes.

A característica forte do SCRUM é o processo rápido: os ciclos duram de sete a 30 dias. Além disso, todo o desenvolvimento cresce de forma incremental, podendo ser alterado durante todo o processo.

O fechamento do projeto baseia-se no ambiente e pode-se dizer que o produto a ser entregue será determinado durante o projeto. A finalização (*post-game phase*) apresenta o produto ao cliente e procura avaliar, com a equipe, a evolução do projeto. Nessa etapa, são tratadas questões como testes, integrações e documentação.

Quer conhecer mais sobre esses modelos?

Acesse na internet o site do Modelo XP *Programming*, que apresenta uma abordagem de programação em pares, com a definição da etapa de testes no início do projeto e intensa participação do usuário final.

Qual é o melhor modelo, qual é o pior? Tais definições dependem da natureza do projeto, da empresa e sua organização, das ferramentas e dos recursos existentes na empresa de desenvolvimento.

Uma ideia que ganha cada vez mais adeptos é a combinação de diferentes modelos, aproveitando-se o melhor de cada um.

Síntese

Nesta primeira unidade você teve contato com conceitos e modelos relacionados ao processo de desenvolvimento de software. Também estudou sobre a importância de se estabelecerem claramente, nas empresas de software, os subprocessos existentes no processo de desenvolvimento.

Foi possível perceber que diferentes modelos servem para o desenvolvimento de sistemas com características específicas. O uso dos modelos também pode ser feito de forma combinada para um mesmo projeto. O projeto pode respeitar o modelo incremental, fragmentando suas funcionalidades; para alguns módulos em que não estão claros os requisitos, pode-se usar a prototipação; para um módulo em que se necessita avaliar algum risco, o modelo espiral é indicado. Assim, misturando-se os modelos, tem-se o melhor de todos eles.

Atividades de autoavaliação

Leia com atenção os enunciados e, após, realize as questões propostas.

1) Classifique as questões a seguir, em Verdadeira (V) ou Falsa (F).

a) ( ) O software pode ser definido como um conjunto de instruções se visto do ponto de vista do programador.

b) ( ) Uma das características mais interessantes do software está relacionada ao fato de ser um produto personalizável. Isso se deve por ser um produto de engenharia e não manufaturado no sentido clássico.

c) ( ) O software e a empresa que o produz são muitas vezes avaliados por apenas uma unidade, pois a mesma pode ser vendida para dezenas de clientes com ou sem customização.

d) ( ) O desgaste do software refere-se apenas à adaptação de produtos a novas tecnologias.

e) ( ) Uma definição completa de software pode ser sugerida a partir da soma de instruções, documentação e estrutura de dados.

2) Relacione a primeira coluna com a segunda.

A. Definição

B. Levantamento de requisitos

C. Desenvolvimento

D. Projeto de software

E. Manutenção

F. Revisões

G. Atividades de apoio

H. Análise de requisitos

( ) Composto por atividades que ocorrem no decorrer de todo o processo de desenvolvimento, como revisões, controle de mudanças e documentação.

( ) Define o que deve ser feito para apoiar o cliente em seus problemas

( ) Nessa etapa, são executados o projeto, codificação e testes do software.

( ) Etapa que identifica as necessidades do cliente.

( ) É a etapa responsável por alterações de cunho corretivo e adaptativo do software.

( ) Atividades efetuadas para garantir que a qualidade seja mantida.

( ) Fazem parte dessa etapa o planejamento do projeto, levantamento e análise de requisitos.

( ) Define como o software irá implementar a solução do cliente.

3) Assinale as afirmativas corretas.

a) ( ) Os métodos proporcionam detalhes relacionados à construção do software para as etapas de planejamento e estimativa.

b) ( ) Os métodos estabelecem detalhes de como fazer para construir um software em todas as etapas do processo; as ferramentas automatizam os métodos, facilitando sua utilização; os procedimentos estabelecem controles, artefatos que assegurem a correta utilização do método.

c) ( ) Procedimentos proporcionam os detalhes de como construir o software; ferramentas automatizam os procedimentos; os métodos formam o elo entre ambos.

4) Relacione as características de cada modelo **[C]**ascata, **[P]**rototipação, **[I]**ncremental ou **[E]**spiral:

a) ( ) Neste modelo de desenvolvimento é possível avaliar riscos de projeto, tomando-se decisões baseadas na experimentação de diferentes soluções.

b) ( ) O modelo facilita a identificação de requisitos para a construção do futuro *software* por meio de protótipos.

c) ( ) O sistema é fragmentado, sendo que cada fragmento percorre todo o ciclo de desenvolvimento do *software*.

d) ( ) Nesse modelo, as subtarefas são executadas de forma sequencial e bastante rígida, sendo que o cliente tem contato com o sistema somente quando ocorre sua conclusão.

5) Assinale com X qual dos modelos a seguir oferece menor contato com o cliente.

a) ( ) Modelo cascata

b) ( ) Modelo incremental

c) ( ) Modelo prototipação

d) ( ) Modelo espiral

6) A empresa CONSTONÓIS procura sua empresa para solicitar o desenvolvimento de um software, para controle de vendas de imóveis. A empresa responsável pela construção de prédios realiza a venda de seus imóveis à vista ou por financiamento direto com a construtora. O sistema deve possibilitar o controle das vendas assim como o controle do financiamento, que aceita ouro, carros, imóveis, consórcios, dólares ou reais como parte do pagamento. O cliente não deixou muito claro o cálculo do saldo devedor do comprador nem a forma como ocorre o acúmulo de débito do cliente. O sistema apresenta, em sua análise inicial, 30 funcionalidades entre cadastros e relatórios de consulta. Qual o modelo de desenvolvimento que você adotaria?

a) ( ) Modelo cascata

b) ( ) Modelo incremental

c) ( ) Modelo prototipação

d) ( ) Modelo espiral

7) Observe o modelo XP e o modelo SCRUM, e a seguir descreva o que é possível determinar como diferenças fundamentais em relação aos modelos tradicionais.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar os seguintes livros:

PRESSMAN, Roger. **Engenharia de Software**. São Paulo: McGraw-Hill, 2002.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. São Paulo: Addison-Wesley, 2003.

UNIDADE 2

2

Engenharia de requisitos

Objetivos de aprendizagem

- Reconhecer a importância da análise de requisitos no processo de desenvolvimento.
- Perceber as diferentes etapas que fazem parte da engenharia de requisitos.
- Realizar de forma consistente a atividade de análise de requisitos.

Seções de estudo

Para início de estudo

Quando você inicia um projeto de *software*, existe sempre uma preocupação: fazer o melhor projeto. Mas qual é o melhor projeto? Na verdade, é aquele que atende o cliente da melhor maneira.

A etapa de engenharia de requisitos é o momento em que as necessidades são exploradas por parte do desenvolvedor em companhia do cliente e sua empresa. Para que essa etapa seja finalizada com uma proposta de solução adequada, é necessário que tarefas, como o estudo de viabilidade, licitação e análise de requisitos, especificação e gerenciamento de requisitos, sejam realizadas de forma cuidadosa e consistente.

Nesta unidade você vai perceber que a engenharia de requisitos é um processo interativo que envolve a compreensão do domínio, a coleta de requisitos, a estruturação e o estabelecimento de prioridades para a execução do projeto.

Você acredita que os requisitos persistem ao longo do tempo? Será que um projeto iniciado em janeiro terá os mesmos requisitos ao final de 12 meses? A compreensão do projetista sobre o problema dispensa a participação do cliente em etapas de verificação? Nos estudos desta unidade, você vai encontrar apoio para responder a algumas dessas questões.

Seção 1 – Engenharia de requisitos

Você já esteve envolvido na construção de uma casa? Ou teve alguém construindo próximo a você? Imagine então que você construirá uma casa e deseja que ela seja espaçosa, com dois pavimentos, três quartos, três banheiros etc. Qual seria a primeira ação a ser tomada? Pense um pouco a respeito.

Bom, a primeira coisa que você com certeza fará é contratar o pedreiro, comprar o material e iniciar a obra, pois você sabe exatamente como sua casa deve ser feita. Não é verdade?

Não? Por quê?

Isso não acontece porque se sabe da necessidade de ter um projeto claro, em projetar os aspectos relacionados à estrutura hidráulica, elétrica, além de ter profissionais aptos a conceber a melhor solução para as nossas suas necessidades.

Depois de finalizado o projeto é que se inicia a construção e, nesse momento, o projeto documentado da planta funciona como um roteiro de gerenciamento, ou seja, tudo o que for construído deve estar de acordo com o que foi projetado.

Isso significa que o que os pedreiros e o mestre de obras realizam com o projeto feito é validado por engenheiros e arquitetos. Se você construísse sem fazer um projeto, acredita que sua casa seria feita dentro de suas expectativas, solucionando suas necessidades? É quase certo que isso não aconteceria!

O exemplo da casa também é válido para a construção de um *software*, mas infelizmente muitas empresas ainda apostam no desenvolvimento do produto sem passar formalmente pela etapa inicial de definição e projeto, indo direto para a programação. O resultado, porém, é muitas vezes desastroso.

Ao se iniciar um projeto de *software*, começa-se uma série de processos, marcados por entregas de artefatos predeterminados pelos modelos de desenvolvimento utilizados e suas metodologias.

A etapa de engenharia de requisitos encontra-se estrategicamente no início do projeto, pois é nessa fase que você vai compreender as necessidades de seu futuro cliente.

Peters (2001) declara com grande propriedade que o grau de compreensibilidade, precisão e rigor da descrição, fornecido por um documento de requisitos de *software*, tende a ser diretamente proporcional ao grau de qualidade do produto resultante.

Mas o que é um requisito?

Requisito de *software* é uma descrição dos principais recursos de um produto de software, seu fluxo de informações, comportamento e atributos.

Pádua (2001) descreve que requisitos são características que definem os critérios de aceitação de um produto. Essas características podem ser divididas em:

- **características funcionais** – que representam o comportamento que o sistema deve apresentar diante das interações do usuário (o cliente deseja que seu vendedor lance o pedido de vendas para emitir relatório de comissões mensais por vendedor);

- **características não funcionais** – que representam aspectos comportamentais do sistema (o cliente deseja que o pedido seja lançado no momento em que é feito pelo cliente, o que pode indicar a necessidade de recursos *wireless*).

Na maioria das vezes, requisitos não funcionais não são solicitados pelo cliente, mas devem ser inerentes ao projeto. São questões como a portabilidade do sistema, sua segurança e confiabilidade dos dados.

Você ainda pode explicar requisitos a partir do conceito de que são formados por:

- **requisitos explícitos** – são as necessidades ou as próprias condições e objetivos propostos pelo cliente (o cliente deseja ter os dados cadastrais de seus fornecedores);

- **requisitos implícitos** – incluem as diferenças entre os usuários, a evolução no tempo, as implicações éticas, as questões de segurança e outras visões subjetivas (o cliente deseja um *site* de comércio eletrônico; questões de segurança talvez não sejam a preocupação dele, por falta de conhecimento em tecnologia, mas são requisitos que devem estar implícitos no seu produto);

- **requisitos normativos** – são decorrentes de normas, leis ou padrões (a emissão de uma nota fiscal deve seguir as regras propostas pela federação).

Na etapa inicial da análise de requisitos, é fundamental que o analista entenda as necessidades do cliente. Isso significa compreender claramente os requisitos explícitos, implícitos e normativos.

Essa compreensão da amplitude do problema é proporcional ao sucesso do projeto com seu cliente final. Parece fácil, mas uma das maiores dificuldades nesse processo é a comunicação entre cliente e desenvolvedor.

Veja a figura a seguir. A visão do que o cliente solicitou ao desenvolvedor, sua necessidade, fica totalmente comprometida por falhas no entendimento e na comunicação. O resultado do "que acaba sendo feito" tem gerado custos elevados para empresas desenvolvedoras e para seus clientes, que em muitas situações não conseguem usar o produto desenvolvido.

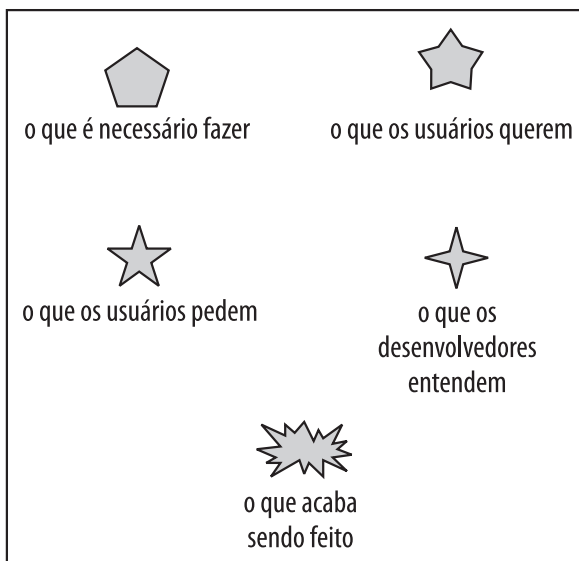



Figura 2.1 – Evolução dos requisitos
Fonte: Pádua (2001).

Lembre-se: A pressa para chegar à implementação não será o caminho mais curto.

A dificuldade de entender o cliente é um desafio que deve ser encarado a partir de técnicas, métodos e **paciência** por parte do desenvolvedor.

Os processos usados durante a engenharia de requisitos variam, dependendo do domínio da aplicação, das pessoas envolvidas e da organização que está desenvolvendo os requisitos.

Existem algumas atividades genéricas comuns a todos os processos, que são:

- levantamento de requisitos;
- documentação de requisitos;
- especificação de requisitos;
- validação de requisitos;
- gerenciamento de requisitos.

Nas próximas seções, você vai estudar sobre as primeiras etapas do processo de engenharia de requisitos, a começar pela etapa de levantamento de requisitos.

Seção 2 – Levantamento de requisitos

É na etapa de levantamento de requisitos que ocorre a compreensão do problema aplicada ao desenvolvimento de *software*.

Quando você está nessa fase, é fundamental que usuários e desenvolvedores tenham a mesma visão do problema a ser resolvido. Isso significa que você precisa conhecer os requisitos tão bem quanto o próprio cliente. Mas, como obter esses requisitos?

Durante o levantamento de requisitos, você vai se deparar com um grande volume de relatórios, formulários e documentos. Mas quais são os que você deve avaliar?

Por outro lado, em uma empresa com 500 funcionários, serão 500 as pessoas envolvidas no processo de levantamento. Você vai entrevistar todas elas?

É nessa hora que você deve perceber a importância de "ver a floresta em vez das árvores". Detecte as pessoas-chave do processo, trabalhe usando amostragens da população. Escute com atenção a gerência da empresa e seus objetivos.

Retome o exemplo da construção de uma casa. Para que o arquiteto inicie o projeto, ele precisa perceber o perfil do cliente, suas preferências e necessidades. Para essa etapa, existem algumas técnicas úteis, como a entrevista, observações, investigação, entre outras.

a) Entrevista

O uso da entrevista é feito pelo uso do formato "pergunta-resposta". Usando essa técnica, você pode obter opiniões do usuário, descobrir o que o cliente pensa sobre o sistema atual, obter metas organizacionais/pessoais e levantar procedimentos informais.

Quando você realizar uma entrevista, lembre-se de:

- tentar estabelecer com o cliente um clima de confiança e entendimento;
- manter-se sempre no controle da entrevista;
- tentar mostrar ao cliente sua importância dentro do sistema;
- preparar-se antecipadamente para a **entrevista**.

Lembre-se: Inclua em sua lista de entrevistados pessoas-chave dentro do futuro sistema.

O que significa preparar-se para a entrevista?

Estude o material previamente, verifique a linguagem utilizada (imagine que, se o *software* for da área jurídica, você precisa falar e entender as nuances e significados dos termos utilizados durante sua entrevista), perceba aspectos relacionados à organização e mesmo sobre o futuro entrevistado. Estabeleça claramente os objetivos, saiba o que perguntar, não faça rodeios.

Quando você realizar uma entrevista, marque a data e a hora com antecedência, com duração de no mínimo 45 minutos e, no máximo, de duas horas. Elabore as questões e a estrutura da entrevista, e durante a entrevista registre tudo o que for possível, fazendo uso de anotações ou de um gravador.

Ao formular as questões, tome algumas precauções.

- Evite usar questões que levem o entrevistado a responder de uma forma específica ou tendenciosa.

Inadequada: "Você também acredita que a prioridade do desenvolvimento deva ser o faturamento, como seu gerente afirmou?"

Melhor: "O que você acha que deve ser implantado em primeiro plano?"

- Fazer duas questões em uma torna a pergunta confusa e a resposta pode não ser completa. Ainda é possível que a pessoa acabe respondendo a uma das questões apenas.

Inadequada: "Em que situações você cancela uma nota fiscal? Quando ocorre o cancelamento de uma nota fiscal, quais os procedimentos?"

b) Questionário

O questionário é uma técnica que permite o levantamento de informações a partir da coleta de informações de diferentes pessoas afetadas pelo sistema.

Segundo Sommerville (2003), nessa técnica são abordadas questões relacionadas ao que as pessoas na organização querem, ao que as pessoas consideram como verdade, ao comportamento das pessoas, às características delas. Procedimentos e equipamentos são levantados.

Um questionário deve ter questões claras e não ambíguas, ter fluxo bem definido, ter administração planejada em detalhes e ainda levantar antecipadamente as dúvidas das pessoas que irão respondê-lo.

Sempre que possível, use o vocabulário das pessoas que irão responder. Prefira perguntas curtas e simples. Certifique-se de que as questões estão tecnicamente precisas antes de incluí-las no **questionário**.

Uma dica interessante é aplicar o questionário em uma amostra de usuários, solicitando a avaliação sobre a estrutura e o conteúdo do mesmo.

c) Observação direta

A observação direta pode ser utilizada como validação das entrevistas, identificação de documentos, esclarecimento sobre o que está sendo realizado no ambiente atual e a forma como ocorre.

No mecanismo de observação direta, o analista observa sem intervir diretamente no processo. É importante planejar a observação e isso significa identificar o que deve ser observado, obter aprovação das gerências apropriadas, obter as funções e nomes das pessoas envolvidas nas ações que serão observadas. Se você optar por essa técnica, prepare os usuários com cuidado, esclarecendo a forma como o processo vai ocorrer.

d) *Brainstorming*

No sentido exato da palavra, *brainstorming* é uma tempestade de ideias. O uso da discussão em grupos, em que a partir dos resultados das técnicas acima procura-se compreender corretamente documentos, respostas oferecidas pelos usuários, processos existentes, é a base para que se chegue a uma boa especificação.

Nessa etapa, inicia-se a formatação de um documento que deve conter os requisitos necessários ao projeto, dentro de um consenso entre desenvolvedores e cliente.

Durante o levantamento dos requisitos, também é estabelecido o escopo do projeto e, ainda, as possíveis restrições que possam delinear algum tipo de risco no horizonte.

Imagine que, durante a entrevista, o cliente diga para você que não pretende investir em equipamentos e em novos produtos, como um banco de dados. Talvez essa informação se torne uma forte restrição para as possíveis soluções e deve, portanto, estar presente na documentação. Como, então, iniciar o levantamento?

Quais pontos você deve levantar?

Peters (2001) sugere o levantamento das seguintes informações:

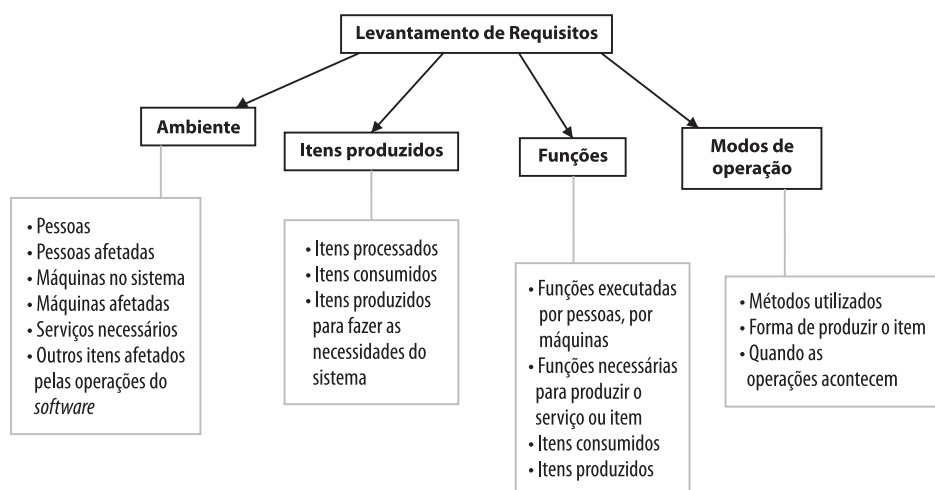



Figura 2.2 – Componentes da análise do problema
Fonte: Peters (2001).

Segundo o autor, o levantamento deve ser feito de forma ordenada, dividido em quatro etapas. Primeiro seriam levantados os requisitos relacionados ao ambiente; depois, às funções existentes; em terceiro lugar, ao modo como as funções são executadas; por fim, os requisitos relacionados aos itens que são inseridos e suas transformações dentro do processo.

Nunca esqueça: nessa etapa avalie os problemas na situação atual. Jamais pense na solução nessa etapa. Concentre-se nas necessidades do cliente.

e) Viabilidade

Antes de você prosseguir, é importante considerarmos um estudo da viabilidade do sistema, se vale a pena ou não sua implementação. Para tanto, é fundamental que esteja claro se o sistema contribui para com os objetivos da organização, se pode ser construído usando-se a tecnologia existente ou, ainda, se o orçamento comporta o que é necessário para sua implementação.

Um fator forte nesse momento de decisões é a possibilidade de integração do sistema aos demais já existentes, se for o caso. Quando você se certifica da viabilidade, está protegendo sua empresa e a empresa do cliente. Esse estudo baseia-se nas informações coletadas durante o levantamento de requisitos.

Sommerville (2003) sugere algumas questões para as pessoas da organização, que podem ser colocadas em um cenário de discussão:

- E se o sistema não fosse implementado?
- Quais são os problemas do processo atual?
- Como o sistema proposto irá ajudar?
- Quais serão os problemas de integração?
- São necessárias novas tecnologias? Em quais habilidades?
- Quais recursos devem ser suportados pelo sistema proposto?

Alguns modelos

Em nossa midiateca estão disponíveis alguns modelos para o levantamento de requisitos que podem servir para futuros levantamentos. Dê uma olhadinha:

Roteiro para Análise do Problema – baseado em Peters (2001).

Seção 3 – Especificação de requisitos

A subetapa de especificação de requisitos visa estabelecer um conjunto de requisitos consistentes e sem ambiguidades que possa ser usado como base para o desenvolvimento do *software*.

Nessa etapa também é comum a negociação para resolver conflitos detectados.

A especificação de um requisito de *software* é a descrição de um produto de software, programa ou conjunto de programa específico que executa uma série de funções do ambiente de destino (IEEE, 1993).

Em resumo, pode-se dizer que **na especificação propomos a solução para o problema do cliente**.

A especificação pode ser totalmente descritiva (como é o caso de alguns documentos da metodologia **RUP** proposta pela *Rational Rose*), ou iniciar a construção de modelos (utilizando-se da notação oferecida pela análise estruturada ou orientada a objetos). Essa decisão depende da metodologia que você estiver utilizando ou mesmo do grau de maturidade da empresa.

O *Rational Unified Process* (RUP) é um processo de engenharia de software que pretende aumentar a produtividade da equipe oferecendo práticas eficientes executadas por meio de diretrizes, templates e orientações sobre ferramentas para todas as atividades críticas de desenvolvimento de software.

O uso de uma especificação textual pode ser feito na forma de relatórios. Peters (2001) sugere um conjunto de quatro questões que devem ser abordadas:

- a primeira delas refere-se às funcionalidades que devem ser definidas e elaboradas, e que o sistema deve suportar;
- a segunda refere-se às interfaces externas do sistema (outros sistemas, equipamentos de *hardware*);
- a terceira relaciona-se ao desempenho esperado pelo produto (questões como tempo de resposta durante uma consulta);
- a quarta relaciona-se a atributos de qualidade (a capacidade do *software* de ser utilizado em diferentes plataformas, por exemplo) e a restrições impostas pelo cliente, pelo ambiente ou pelo próprio desenvolvedor.

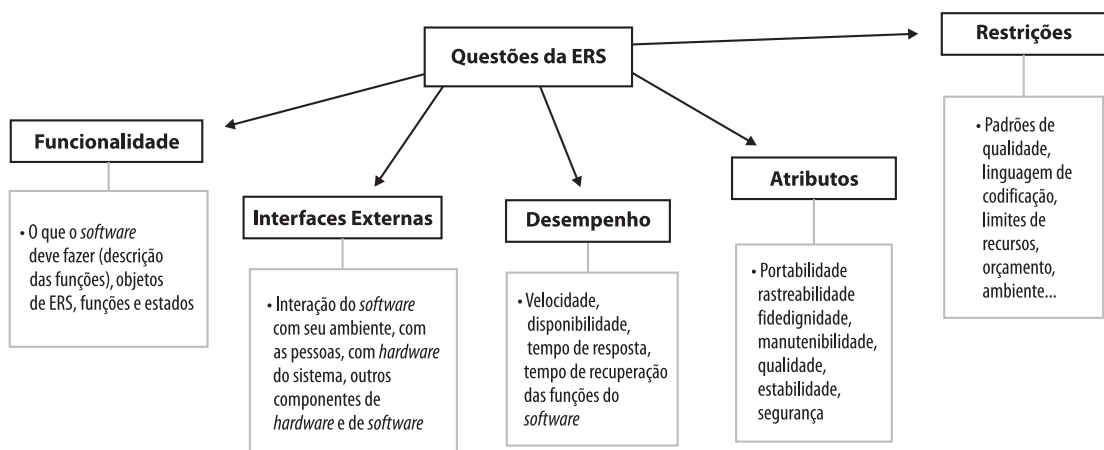



Figura 2.3 – Questões básicas ao se escrever uma ERS
Fonte: Peters (2001).

A especificação dos requisitos faz parte da documentação do sistema, e pode-se dizer que é um de seus principais artefatos. Sabendo disso, é necessário decidir "o que" deve ser documentado sobre essa etapa. Peters (2001) sugere uma estrutura composta de quatro pontos principais:

- Introdução (onde são descritas características do documento);
- Descrição global (onde são referenciados perspectivas do produto, riscos associados e funcionalidades requeridas);

- Requisitos específicos (apresenta um esqueleto de interfaces e necessidades que devem ser suportadas pelo produto em termos de atributos e performance);
- Rastreabilidade dos requisitos (que incluem casos de teste para a validação do futuro projeto).

1. Introdução	
1.1 Objetivo 1.2 Escopo 1.3 Definições, acrônimos e abreviações 1.4 Referências 1.5 Visão geral	
2. Descrição global	
2.1 Perspectiva do produto 2.2 Funções do produto 2.3 Características do usuário 2.4 Restrições 2.5 Hipóteses e dependências 2.6 Distribuição de requisitos	Perspectiva do produto: seu relacionamento com o sistema maior. Hipóteses: indicam como alterações feitas na ERS afetam seções específicas da ERS (exemplo: quais diagramas de fluxo de dados são afetados ou mudanças de funções ou exclusão das mesmas)
3. Requisitos específicos	
3.1 Interfaces externas 3.2 Requisitos de processo e dados 3.3 Requisitos de desempenho e qualidade 3.4 Requisitos de banco de dados lógico 3.5 Restrições de projeto 3.6 Atributos de sistema de software 3.7 Organização de requisitos específicos	Restrições de projeto: políticas regulamentares, limitações de *hardware*, interfaces com outros aplicativos, operação paralela, segurança e perfil etc.
4. Rastreabilidade dos requisitos	
Apêndice	
Índice remissivo	

Quadro 2.1 - Estrutura para documentação do sistema
Fonte: Elaboração da autora (2008).

Quer conhecer mais?

Se você estiver interessado nesse modelo oferecido por Peters, leia o capítulo 2 do livro:

PETERS, J. F.; PEDRYCZ, W. **Engenharia de software**: teoria e prática. Rio de Janeiro: Campus, 2001.

Modelos

A especificação também pode ser feita na forma de modelos. Mas você sabe o que é um modelo?

Um modelo é uma representação de alguma coisa do mundo real, uma abstração da realidade. Além disso, tem a finalidade de servir como fundamento para o projeto de *software*, facilitando a compreensão do fluxo de dados e de controle, do processamento funcional, da operação comportamental e do conteúdo da informação.

Nas primeiras etapas do processo de desenvolvimento (levantamento de requisitos e análise), o engenheiro de *software* representa o sistema por meio de modelos conceituais, considerando características do sistema independentemente do ambiente computacional (*hardware* e *software*) no qual o sistema será implementado nas etapas posteriores.

Na etapa seguinte, as características lógicas (características dependentes de um determinado tipo de sistema computacional) e físicas (características dependentes de um sistema computacional específico) são representadas em novos modelos.

A representação dos modelos lógicos e físicos pode ser feita por meio da análise estruturada, da análise essencial ou da análise orientada a objetos. Os diagramas utilizados nessas metodologias apoiam e facilitam o entendimento da solução.

Visite o EVA, ferramenta Midiateca. Lá você encontrará modelos para a especificação de requisitos sem a utilização de modelos de análise totalmente textuais. Leia sobre:

Roteiro para Especificação de Requisitos – baseado em Peters (2001).

Software Requirements Specification (without Use-Case) da metodologia RUP – Rational Unified Process.

É importante salientar: Ao finalizar a especificação, solicite a assinatura do cliente. Essa assinatura faz com que sua especificação valha como um contrato.

Seção 4 – Atividades de apoio

As atividades de documentação, verificação, validação e gerenciamento não são exclusivas da etapa de análise de requisitos às atividades de apoio, mas estarão presentes em todo o processo de desenvolvimento do *software*. Dentro do processo de análise de requisitos, essas atividades não ocorrem de forma obrigatoriamente sequencial, mas sim de forma paralela.

a) Documentação de requisitos

É a atividade de representar os resultados da engenharia de requisitos em um documento, contendo os requisitos do *software*.

b) Verificação e validação de requisitos

Verificar e validar os requisitos é uma etapa do processo que não pode ser menosprezada. Na verificação de requisitos, você avalia se a especificação de requisitos está sendo construída de forma

correta, seguindo os padrões previamente definidos, evitando requisitos confusos, duplicados, incoerentes ou incompletos.

A validação verifica se os requisitos e modelos documentados são o reflexo das reais necessidades e requisitos do cliente.

Se você passar por essa etapa em seu projeto, vai evitar a descoberta de interpretações errôneas ou mesmo deficiências do projeto em etapas futuras. Isso significa uma economia de tempo e dinheiro significativa.

O que você deve observar na validação?

Sommerville (2003) sugere um check-list. Observe:

- O sistema fornece as funções que melhor apoiam as necessidades do usuário?
- Há algum conflito nos requisitos?
- Todas as funções exigidas pelo cliente foram incluídas?
- Os requisitos podem ser implementados com o orçamento e a tecnologia disponíveis?
- O requisito foi apropriadamente compreendido?
- A origem do requisito foi claramente estabelecida?
- O requisito pode ser alterado sem um grande impacto em outros requisitos?

Existem algumas técnicas que apoiam a validação de requisitos:

- As revisões de requisitos pela análise sistemática e regular dos requisitos.

- O uso da prototipação para validar entradas e saídas, por exemplo, por meio de protótipos não funcionais de telas e relatórios (procure envolver equipe e cliente nessas revisões).
- A geração de casos de teste para os requisitos, verificando se é possível construir os casos de teste e mesmo testar aquele requisito.

c) Gerenciamento de requisitos

A tarefa de gerenciar requisitos se preocupa com as mudanças nos requisitos que já haviam sido acertadas entre cliente e desenvolvedor.

Em outras palavras, é preciso: documentar essas mudanças, gerenciar os relacionamentos entre os requisitos e suas dependências que podem afetar outros requisitos e outros artefatos, produzidos no processo de *software*; verificar a credibilidade da solicitação de mudança com a empresa.

O gerenciamento deve manter as alterações de requisitos de forma controlada, tornando as alterações sustentáveis durante o desenvolvimento.

Fazer mudanças nos requisitos não deve ser uma catástrofe: é comum a dificuldade de reconhecer todos eles em uma etapa inicial. O importante é documentar, relacionar, controlar essas mudanças, repassando-as a todo o processo de desenvolvimento.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize as atividades propostas.

Síntese

O objetivo do desenvolvedor de *software* sempre deve ser o fornecimento de um *software* de qualidade que atenda às necessidades dos clientes. Nesta unidade você aprendeu que, para obter essa qualidade, é preciso realizar a etapa de análise de requisitos de forma consistente, utilizando metodologias apropriadas que permitam a revisão constante de todo o processo. O uso de diferentes técnicas, como entrevistas e questionários durante o levantamento de requisitos, ajuda a variar o foco da observação do projetista. Na especificação, o uso de diferentes modelos pode ser adaptado de acordo com as características da empresa. As atividades de apoio, como o gerenciamento de requisitos, permitem rastrear alterações de requisitos e comunicar essas alterações a toda a equipe envolvida.

A engenharia de requisitos é um momento de conhecimento, reconhecimento e de explosão de ideias. O bom aproveitamento dessa etapa é fundamental; estudos demonstram que o custo de alterações em etapas posteriores à análise de requisitos pode chegar a 100% do custo original do projeto.

Nesta unidade, você também percebeu que a engenharia de requisitos incorpora o uso de modelos. O uso de modelos permite desenvolver o *software* de maneira previsível em um determinado período, com utilização eficiente e eficaz de recursos. O modelo ajuda a visualizar o sistema como desejamos, simplificando seu entendimento.

Atividades de autoavaliação

Leia com atenção os enunciados e, após, realize as questões propostas.

1) Quanto ao requisito, é correto afirmar:

a) ( ) Um requisito é expresso por suas características funcionais.

b) ( ) O requisito de software pode ser expresso por características implícitas, explícitas e normativas.

c) ( ) As características normativas de um requisito estão relacionadas às suas características comportamentais.

d) ( ) O requisito mais importante em uma análise é o requisito explícito.

2) Na análise de requisitos, temos as seguintes etapas:

(a) levantamento

(b) especificação

(c) validação

(d) gerência

Relacione as descrições abaixo com a etapa correspondente.

a) ( ) Responsável pelo controle de alterações de requisitos.

b) ( ) Utiliza-se de técnicas, como observação direta, entrevistas e questionários, para apurar informações pertinentes ao processo.

c) ( ) Avalia se a especificação de requisitos está seguindo os padrões previamente definidos, evitando requisitos confusos, duplicados, incoerentes ou incompletos.

d) ( ) Responsável pela definição de restrições, funcionalidades, atributos, interfaces externas e desempenho.

3) No texto abaixo você tem a solicitação para desenvolvimento de software de um cliente proprietário de uma clínica médica. A partir do texto levantado com o cliente, preencha o documento de análise do problema que dará início à documentação do futuro sistema.

Empresa : Clínica Bem-Estar

1) Função: fomos contratados para analisar seu processo atual e verificar como expandir suas operações e melhorar seu nível de serviço.

Histórico:

A clínica, fundada há 5 anos, atua no atendimento clínico pediátrico.

A clínica possui 34 médicos cadastrados em diferentes especialidades como: cardiologia, clínica geral, dermatologia etc. Todos os médicos utilizam internet e e-mail. A faixa etária predominante é de 30, 35, 40, 42, 44 e 48 anos. Todos os médicos são aptos do ponto de vista físico.

O paciente pode ser atendido de forma particular ou por convênios. Os convênios atendidos são o Bruxtr, Vpfzm e UIOlk.

Cada médico faz 3 plantões semanais de 4 horas seguidas; as consultas possuem um intervalo de 30 minutos. Existe a possibilidade de a consulta ser de retorno, nesse caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde são marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio. Trabalham há 3 anos na clínica com planilhas Excel.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente, que contém nome, endereço, telefone, data de nascimento, convênio.

O médico, ao atender o paciente, preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

A clínica possui de 700 a 800 fichas, sendo que cerca de 600 são de atendimento por convênio.

O gerente da clínica está ansioso, pois não consegue controlar questões relacionadas ao número de pacientes atendidos por convênio e particular, médicos mais procurados e picos de movimento.

Volume de atendimentos: 56 por dia.

Outra questão de interesse é manter um controle de laboratórios conveniados, pois o médico poderia indicar o laboratório já no momento da prescrição.

TEMPLATE PARA REALIZAÇÃO DA ANÁLISE DO PROBLEMA CLÍNICA BEM-ESTAR

1) Nome da empresa ____

2) Contato ____

3) Descrição do problema

____

____

____

____

4) Identificação do principal objetivo do cliente

____

____

____

5) Descrição dos usuários do sistema (para cada usuário descreva cargo e possíveis funções dentro do processo)

____

____

____

6) Descrição detalhada dos processos existentes (COMO O SISTEMA ATUAL FUNCIONA?)

Para marcação da consulta

____

____

____

Saiba mais

Para aprofundar as questões abordadas nesta unidade, pesquise em:

SOMMERVILLE, Ian. **Engenharia de** *software*. 6. ed. São Paulo: Addison-Wesley, 2003.

UNIDADE 3

3

Análise estruturada

Objetivos de aprendizagem

- Reconhecer objetivos e características inerentes ao uso da modelagem estruturada.
- Fazer uso de conceitos e diagramas da modelagem estruturada.
- Compreender e reconhecer uma estrutura que se utilize da modelagem estruturada.
- Empreender o uso da modelagem estruturada.

Seções de estudo

Para início de estudo

Quando se inicia o processo de desenvolvimento de um *software*, são realizadas tarefas específicas que podem ser estruturadas dentro de um modelo de desenvolvimento (como você estudou na primeira unidade da disciplina).

No início do processo tem-se a etapa de engenharia de requisitos (assunto estudado na segunda unidade), na qual você é confrontado com as reais necessidades de seu cliente.

Ainda que você esteja ansioso por começar o processo de desenvolvimento e efetivamente iniciar a etapa de codificação, a etapa crucial do processo ainda está por vir: o projeto de desenvolvimento de *software*.

O projeto está contido na etapa de desenvolvimento do *software*, e é nela que você vai desenhar o modelo da solução que deve resolver as necessidades apontadas no levantamento de requisitos. Existem diferentes metodologias para cumprir essa etapa, mas as mais conhecidas são a análise estruturada e a análise orientada a objetos.

Na análise estruturada, o desenvolvimento do sistema é voltado para as funções (processos) que ele deve realizar. Nesta unidade, você conhecerá as notações e características específicas utilizadas em uma modelagem estruturada e que são fundamentais em sua documentação.

Seção 1 – Análise estruturada

A análise estruturada foi desenvolvida nos anos 1970 por Gane, Sarson e De Marco. A técnica é baseada na utilização de uma linguagem gráfica para construir modelos de um sistema, incorporando também conceitos relacionados às estruturas de dados.

Yourdon (1992) discorre sobre as características da análise estruturada: Nesta técnica, é preciso ter em mente que todo o desenvolvimento do sistema é voltado para as funções que o sistema deve realizar.

O que são essas funções? Imagine o sistema de uma videolocadora. Nele você tem cadastro de clientes, cadastro de DVDs, relatório de clientes da locadora, entre outras funções que o sistema deve executar. As funções utilizam e produzem informações para o ambiente (na função cadastro do DVD, o atendente faz a entrada de dados, digitando informações sobre o DVD, como nome do filme, dos atores, do tipo de filme etc.). Essas informações são repassadas às entidades externas (que pode ser o cliente ou o atendente da locadora) por meio de fluxos de dados ou armazenadas em depósitos de dados (são os depósitos que vão armazenar em seu HD as informações do cliente, dos DVDs etc.).

Retome o exercício da Clínica Bem-Estar, apresentado na unidade anterior.

Empresa: Clínica Bem-Estar

1) Função: fomos contratados para analisar seu processo atual e verificar como expandir suas operações e melhorar seu nível de serviço.

Histórico:

A clínica, fundada há 5 anos, atua no atendimento clínico pediátrico.

A clínica possui 34 médicos cadastrados em diferentes especialidades como: cardiologia, clínica geral, dermatologia etc. Todos os médicos utilizam internet e e-mail. A faixa etária predominante é de 30, 35, 40, 42, 44 e 48 anos. Todos os médicos são aptos do ponto de vista físico.

O paciente pode ser atendido de forma particular ou por convênios. Os convênios atendidos são o Bruxtr, Vpfzm e UIOlk.

Cada médico faz 3 plantões semanais de 4 horas seguidas; as consultas possuem um intervalo de 30 minutos. Existe a possibilidade de a consulta ser de retorno, nesse caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde são marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio. Trabalham há 3 anos na clínica com planilhas Excel.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente, que contém nome, endereço, telefone, data de nascimento, convênio.

O médico, ao atender o paciente, preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

A clínica possui de 700 a 800 fichas, sendo que cerca de 600 são de atendimento por convênio.

O gerente da clínica está ansioso, pois não consegue controlar questões relacionadas ao número de pacientes atendidos por convênio e particular, médicos mais procurados e picos de movimento.

Volume de atendimentos: 56 por dia.

Outra questão de interesse é manter um controle de laboratórios conveniados, pois o médico poderia indicar o laboratório já no momento da prescrição.

Se você fosse listar alguns dos requisitos funcionais, certamente listaria:

- Cadastro de médicos que possibilite inserir, alterar e excluir os dados dos médicos da clínica.
- Cadastro de pacientes que permita inserir, alterar e excluir os dados dos pacientes da clínica.
- Agenda médica onde deve ser possível o agendamento, consulta e exclusão do agendamento de consulta de um paciente, com um determinado médico, em um determinado horário.
- Ficha médica que permita o lançamento e consulta de dados sobre a consulta do paciente.

- Emissão de relatório estatístico sobre o atendimento de convênios.
- Emissão de relatório estatístico sobre o atendimento por médicos.
- Emissão de relatório estatístico sobre o número de atendimentos durante o ano.
- Cadastro de laboratórios conveniados à clínica.

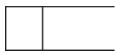
Se fôssemos listar os requisitos não funcionais:

- O sistema deve operar de forma confiável no lançamento de todos os dados de pacientes e consultas e sua gravação.
- O sistema deve proteger o acesso às informações históricas da ficha do paciente, sendo que seu acesso deve ser exclusivo para a equipe médica
- O sistema deve ser desenvolvido sobre uma plataforma *open source* permitindo portabilidade.
- O sistema deve seguir requisitos de usabilidade para as interfaces, sendo de fácil aprendizado, pois a rotatividade das atendentes é grande.

A partir da lista de requisitos, você pode iniciar o processo de modelagem. Para tanto, é necessário entender alguns conceitos básicos.

A notação da análise estruturada é composta pelos seguintes elementos:

- Diagrama de Fluxo de Dados (DFD);
- dicionário de dados;
- especificação dos processos;
- modelagem de dados (ER).

Nas próximas seções, você conhecerá detalhadamente cada um desses elementos.

Seção 2 – Diagrama de Fluxo de Dados (DFD)

Segundo Gane (1983), um Diagrama de Fluxo de Dados (DFD) é uma técnica gráfica de representação que permite explicitar os fluxos de informação e as transformações que são aplicadas, à medida que os dados se deslocam da entrada em direção à saída.

O DFD permite particionar o sistema, demonstrando seus componentes ativos (como usuários, equipamentos, arquivos) e o interfaceamento de dados que ocorre entre eles.

Para realizar um DFD, são necessários alguns símbolos que o compõem. Conheça cada um:

Símbolo	Descrição
Processo	■ A bolha ou círculo que representa o processo. O processo é um executor de tarefas – funções, atividades. Representa tarefas de processamento do sistema. ■ Quando você colocar nome em um processo, procure utilizar um nome que descreva **o que** o processo faz. ■ Um exemplo de processo pode ser: Cadastrar Paciente, Cadastrar Médico, Agendar Consulta.
Entidades externas	■ O retângulo representa entidades externas (EE), que representa origens e destinos dos dados que percorrem o sistema. Quando você pensar em entidade externa, pense que são criadores e/ou consumidores de dados/informação. ■ Uma EE pode ser uma pessoa, organização ou outros sistemas que interagem com o sistema para envio e/ou recebimento de informações. Pode representar a interface entre o sistema e o mundo externo. ■ A EE pode ser então, no caso da Clínica Médica, o Paciente, o Atendente, o Médico. Se na clínica houver um laboratório que de alguma maneira receba dados da clínica por meio do sistema, então, neste caso, o Laboratório pode ser considerado uma EE.
Depósito de dados	■ A caixa aberta, ou linha dupla preenchida com um rótulo, representa o depósito de dados. Na verdade, são os locais onde ocorre o armazenamento de dados, é um repositório de dados (temporários ou permanentes). ■ Todas as informações do paciente como nome, endereço, telefone serão armazenados em seu HD em um depósito de dados. Neste exemplo, um nome adequado para o depósito seria Paciente.

Fluxo de dados	■ O fluxo de dados é expresso por setas rotuladas que interligam os processos e que permitem indicar o caminho seguido pelos dados. O fluxo de dados realiza a comunicação entre os processos, os depósitos e as entidades externas. ■ O fluxo de dados não significa sequência de um módulo de programa para outro. Mas a seta indica o caminho pelo qual uma ou mais estruturas de dados deverão passar. ■ Quando estamos realizando o processo Cadastrar Paciente, deve existir um fluxo de dados, em que são repassadas do paciente ao processo Cadastrar Paciente as suas informações cadastrais (nome, endereço etc.).

Quadro 3.1 – Símbolos que compõem o DFD
Fonte: Elaboração da autora (2008).

Acompanhe na figura 3.1, a seguir, o processo Cadastrar Paciente.

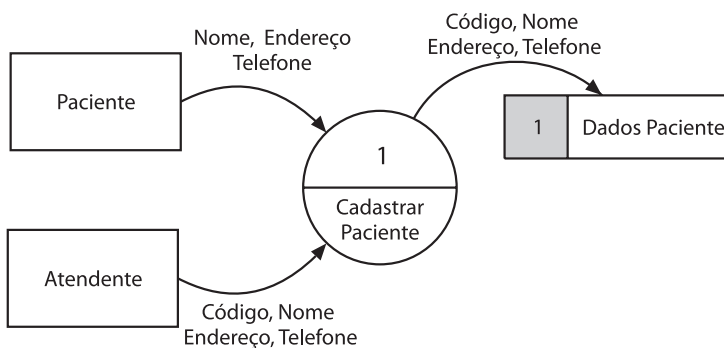



Figura 3.1 – DFD (processo Cadastrar Paciente)
Fonte: Elaboração da autora (2008).

São possíveis perguntas que você pode realizar:

– Por que Cadastrar Paciente (círculo) está representado como um processo?

Resposta: Ele é um processo porque representa uma execução, uma função que deve ser executada no sistema para que o usuário alcance seu objetivo, que é fazer o cadastro do paciente para a clínica.

– Quem são as entidades externas (retângulo) para esse processo?

Resposta: São todos os consumidores e criadores das informações que esse processo vai gerar. Observe que, para cadastrar um paciente, é necessário que o paciente informe seus dados. Além dele, temos a atendente que vai inserir os dados do paciente no sistema. Assim, esse processo tem no mínimo duas entidades externas: Paciente e Atendente.

– Por que Dados Paciente é um depósito de dados no diagrama?

Resposta: Todas as informações do paciente devem ser armazenadas no sistema (se não fosse assim, o paciente teria de informar seus dados a cada consulta na clínica).

Para representar a armazenagem dos dados, a notação usada é o depósito de dados. Portanto, nesse processo (figura 3.1) os dados do paciente representam um depósito de dados em que as informações do paciente serão "guardadas" pelo sistema na clínica.

Seria assim: a entidade externa informa seus dados para o processo Cadastrar Paciente (nome, endereço e telefone), a Atendente (entidade externa), por sua vez, informa os dados ao processo Cadastrar Paciente (por meio do fluxo de dados). Após alimentarem o processo, essas informações são transformadas e os dados do paciente são armazenados no depósito de Dados Paciente.

Observe que na figura 3.1 há um número 1 no processo Cadastrar Paciente e um número 1 no depósito de Dados Paciente. Esses números são utilizados apenas para organizar o DFD para facilitar sua leitura. Na verdade, são sequenciais e sua utilização é opcional.

Na documentação, você pode se referir ao processo por seu número não sendo necessário escrever todo o nome do processo.

Quando se elabora um DFD, faz-se necessária a observação de algumas diretrizes.

Lembre-se de que, quando usamos fluxos de dados, estamos representando o deslocamento de informações. Isso pode acontecer **somente** entre:

- um processo e uma entidade externa;
- dois processos;
- um processo e um depósito de dados.

Quando você nomear o processo, procure utilizar verbos no infinitivo. As entidades externas, os fluxos de dados e os depósitos devem ser identificados por substantivos.

Existe um fator muito importante em um DFD: ele não é um fluxograma, portanto ele jamais deve representar a sequência em que os processos são ativados.

O DFD que é apresentado na figura 3.2 é bastante genérico e pode deixar margens a dúvidas. Quando isso acontece, é possível "explodir" o DFD em níveis de profundidade diferentes. A decisão do número de níveis necessários depende da complexidade do projeto: quanto maior o número de níveis, melhor a descrição do comportamento do sistema.

Mas como ocorre essa explosão em níveis?

Os níveis podem ser descritos como:

a) **Nível 0 (Topo)** – também chamado de **Diagrama de Contexto.** É formado por um único processo que representa o sistema inteiro.

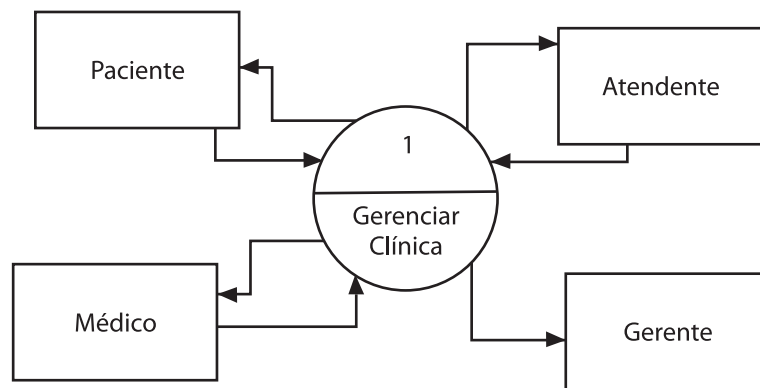



Figura 3.2 – DFD – Nível 0
Fonte: Elaboração da autora (2008).

b) **Nível n (Folhas):** neste caso, os processos são primitivos, sua descrição fica em um nível alto, pois as funções são simples, com poucas E/S.

Nível 1 a n-1 (Intermediários): neste caso, os processos são decompostos em níveis. O n+i é a decomposição/particionamento de processos, fluxos e depósitos do nível i. Em outras palavras: você pode decompor o DFD de forma granular, chegando a um detalhamento minucioso.

Veja o exemplo do DFD Agendar Consulta. Ele pode ser apresentado em um nível apenas:

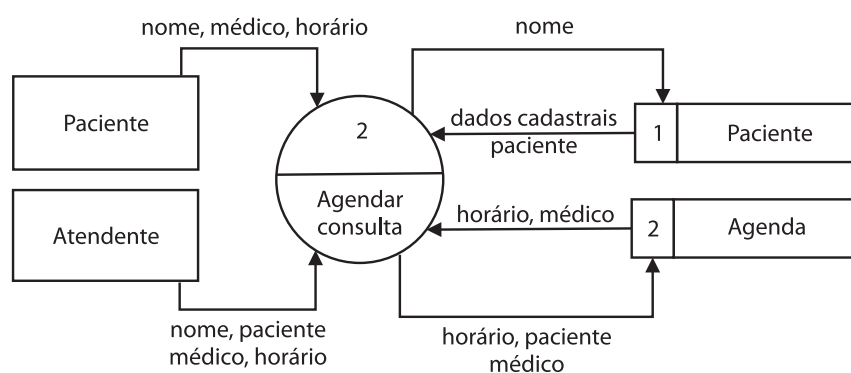



Figura 3.3 – Agenda Consulta Nível 1
Fonte: Elaboração da autora (2008).

Mas, se você quiser detalhar esses processos, o DFD oferece mais informações para o desenvolvedor:

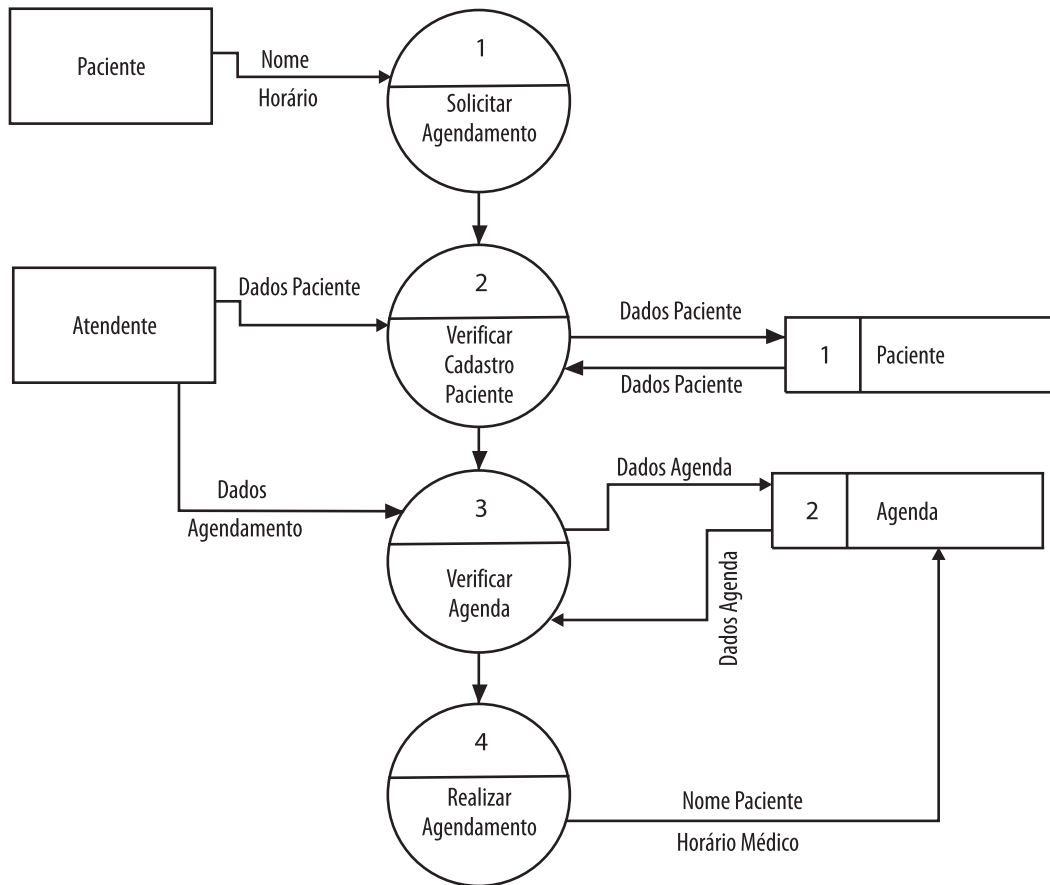



Figura 3.4 – DFD – Agendar Consulta
Fonte: Elaboração da autora (2008).

A figura 3.4 é um exemplo da explosão do DFD Agendar Consulta. Agora já é possível perceber que existem pelo menos quatro subprocessos envolvidos no agendamento da consulta:

- solicitar agendamento (disparado pela entidade externa – EE paciente).
- verificar cadastro paciente (onde será verificada a existência ou não de cadastro do paciente na clínica médica).

- verificar agenda (nela será verificado o horário solicitado, o médico e suas disponibilidades para a consulta).
- realizar agendamento (processo que efetiva a marcação da consulta, armazenando a informação no depósito de dados).

Dicas para a construção de DFDs

Os DFDs são construídos a partir da lista de funcionalidades propostas para o sistema.

a) Escolha nomes significativos para os processos, os fluxos, os depósitos e as entidades externas. Os nomes devem ser facilmente identificados, principalmente pelo vocabulário do usuário.

b) Numere os processos. Em um sistema grande, a numeração pode ser muito útil.

c) Sempre tenha em mente a facilidade de entendimento do DFD. Se o DFD estiver confuso, refaça-o até que lhe pareça suficientemente claro.

A figura 3.5 apresenta um DFD que representa os processos a serem realizados para o controle e faturamento de um sistema de pedidos.

Observe que nesse caso temos três processos envolvidos para a realização do pedido: dois depósitos de dados (Faturas e Pedidos) e apenas uma entidade externa (Clientes). A comunicação entre processos evolui pela passagem de dados por meio dos fluxos de dados.

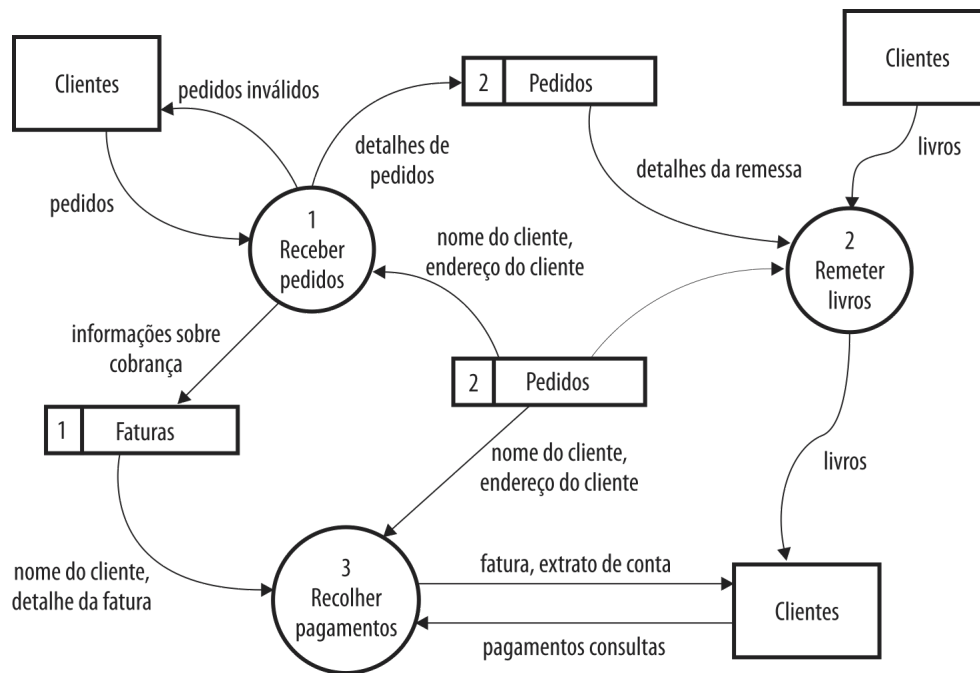



Figura 3.5 – DFD Controle de Pedidos
Fonte: Yourdon (1992).

Observe a seguinte situação: você recebe a missão de desenvolver a modelagem estruturada de um sistema o qual se pretende o desenvolvimento de um controle de pedidos para uma floricultura atacadista. A empresa em questão faz a revenda de flores e folhagens para floriculturas da cidade não tendo venda a varejo.

Os requisitos funcionais da floricultura são:

- manter o cadastro de seus produtos à venda (a venda é exclusivamente de flores e folhagens);
- manter o cadastro de seus clientes;
- manter o seu controle de pedidos; e
- controlar o pagamento de pedidos realizado por seus clientes.

O requisito não funcional da floricultura é:

- a manutenção da simplicidade do código permitindo a facilidade na manutenção.

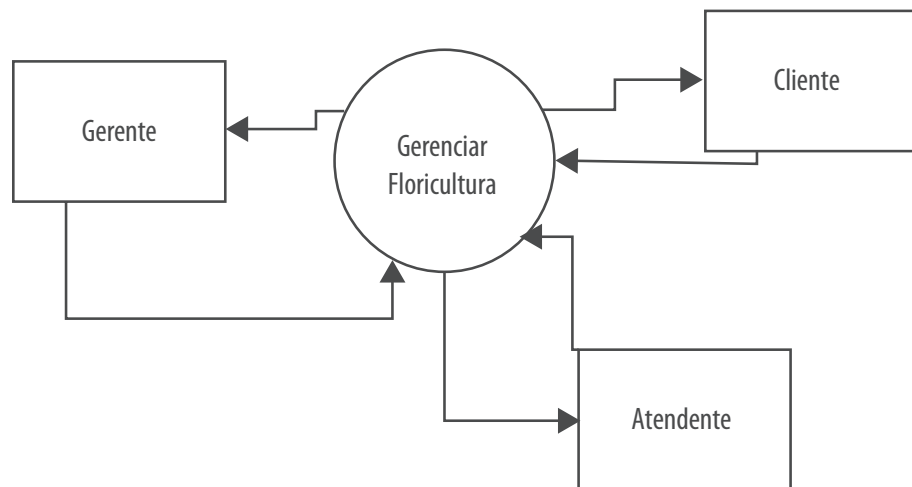



Figura 3.6 – DFD Nível 0 – Sistema Floricultura
Fonte: Elaboração da autora (2008).

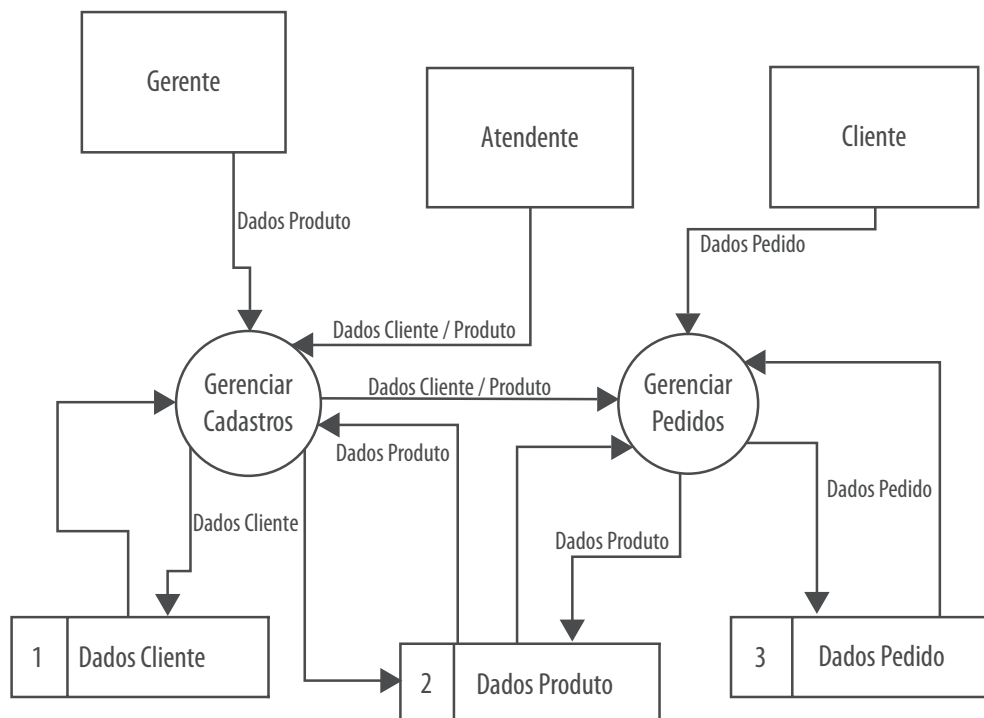



Figura 3.7 – DFD Nível 01 – Sistema Floricultura
Fonte: Elaboração da autora (2008).

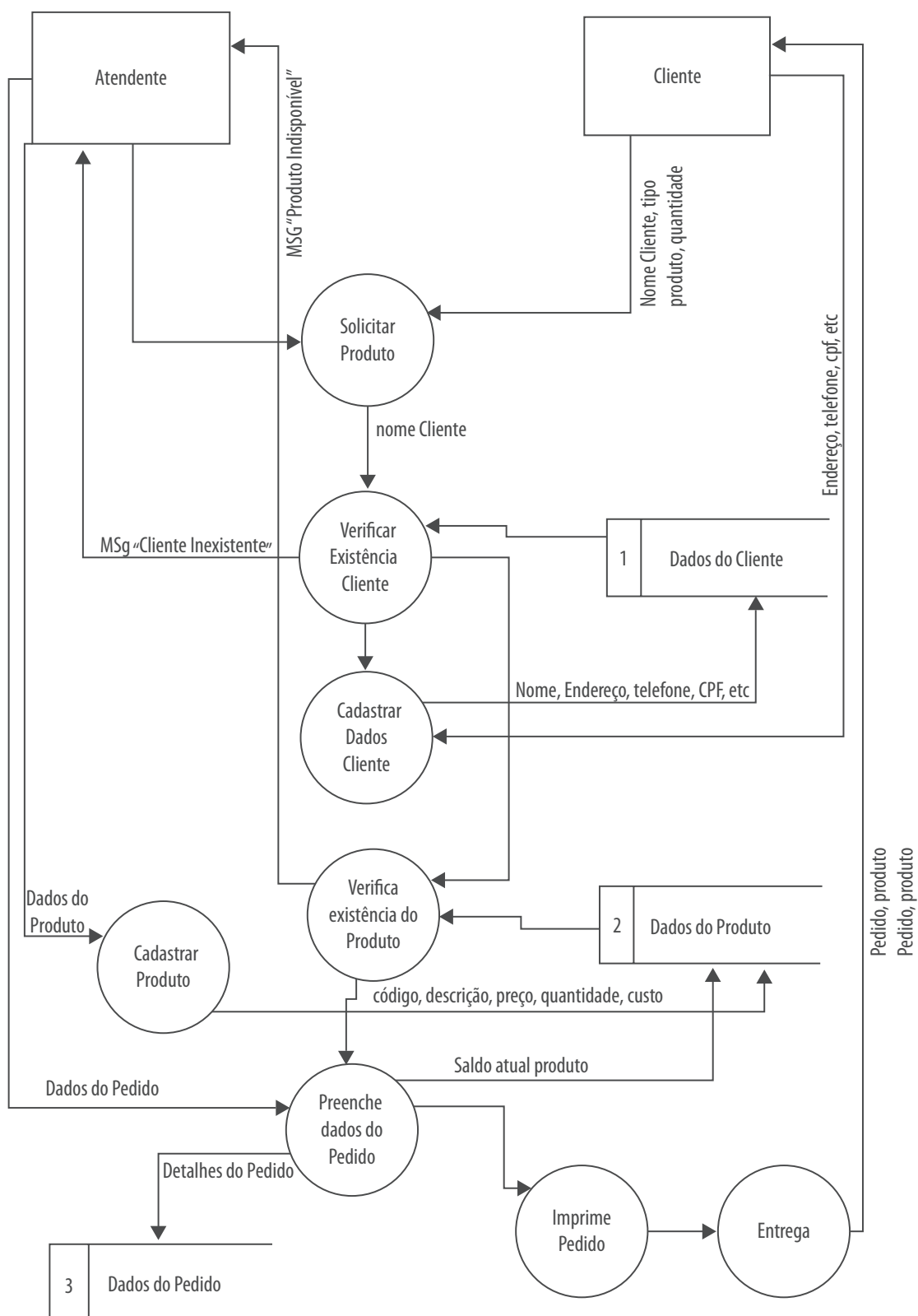



Figura 3.8 – DFD Nível 2 – Sistema Floricultura
Fonte: Elaboração da autora (2008).

Seção 3 – Dicionário de dados

O dicionário de dados é uma descrição precisa de todas as informações presentes no DFD para que usuários e analistas possam compreender todo o modelo de forma simplificada.

Quando se constrói um DFD, os componentes são identificados por um rótulo único.

Apenas o rótulo apresentado no DFD não define a estrutura dos dados que representarão a informação. Essa definição é feita no dicionário de dados.

Mas o que deve ser descrito no depósito de dados?

Pode-se definir o significado de depósitos, a descrição dos fluxos de dados, a estrutura dos arquivos e a especificação lógica dos processos.

As notações mais diversas podem ser utilizadas para definir as estruturas dos dados, sendo que uma proposta de notação possível, de acordo com Mazzola (2011), inclui as seguintes informações:

- **nome** – o identificador principal do item de dados, do depósito de dados ou de uma entidade externa; e, eventualmente, outros nomes utilizados para o mesmo item;

- **especificação** – numérico, alfanumérico, data, inteiro, tamanho máximo, formatação, domínio;

- **utilização –** em que contexto (onde e como) o item de informação é utilizado;

- **descrição –** uma notação que permita explicitar o conteúdo do item;

- **informações complementares** a respeito do item de dados, como valores iniciais, restrições etc.

Observe o exemplo de um dicionário de dados para o depósito de dados Paciente (figura 3.9).

Depósito

Nome do depósito: Paciente.
Especificação: banco de dados cadastrais do paciente, volume aproximado 3500 registros.
Descrição: o depósito de dados Paciente deve armazenar todos os dados cadastrais do paciente da clínica tendo como chave o nome do paciente.
Utilização: o depósito será usado no processo Cadastrar Paciente, Agendar Consulta.

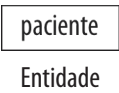
Atributos do depósito de Dados Paciente:

Nome do atributo: Nome_Paciente.
Especificação: campo alfanumérico, tamanho 50, chave primária do depósito de dados.
Descrição: nome do paciente.

Nome do atributo: Endereço_Paciente.
Especificação: campo alfanumérico, tamanho 60 caracteres.
Descrição: endereço do paciente.

Nome do atributo: Telefone_Paciente.
Especificação: campo alfanumérico, tamanho 12.
Descrição: telefone do paciente.

Nome do atributo: Sexo_Paciente
Especificação: campo alfanumérico, tamanho 1, assume os valores “F” ou “M”.
Descrição: sexo do paciente.

Figura 3.9 – Exemplo de dicionário de dados para o depósito de dados do Paciente
Fonte: Elaboração da autora (2008).

Seção 4 – Modelagem de dados

A modelagem de dados é também conhecida como diagrama ER (Entidade-Relacionamento). Essa técnica foi desenvolvida originalmente para dar suporte ao projeto de bancos de dados (CHEN, 1990).

O modelo ER é uma técnica utilizada para representar os dados a serem armazenados em um sistema.

A simbologia utilizada em um diagrama ER é composta pelos seguintes elementos:

paciente Entidade	Uma entidade representa um objeto concreto ou abstrato em que serão armazenadas as informações. Uma entidade pode ser uma pessoa, uma instituição, elementos do domínio da aplicação (Paciente, Agenda). Quando você cria uma entidade, ela é composta por um conjunto de instâncias. Cada instância é uma única ocorrência de uma determinada entidade. Em um depósito Paciente, João Dirceu é uma instância de Paciente.
Atributo	Os atributos são utilizados para descreverem características ou propriedades elementares de entidades e relacionamentos. Os atributos representam o conteúdo de uma entidade. Pensando no exemplo da clínica, alguns atributos são NomePaciente, TelefonePaciente, SexoPaciente etc.
Relacionamento	Um relacionamento é uma abstração de uma associação entre duas ou mais entidades. Pode haver mais de um relacionamento entre objetos. Um exemplo de relacionamento ocorre quando pensamos que o paciente Almir (uma instância da entidade Paciente) possui "n" agendamentos (na entidade Agenda).

Quadro 3.2 – Simbologia utilizada em um diagrama ER
Fonte: Elaboração da autora (2008).

– Imagine o sistema da clínica. Quais entidades você consegue identificar?

Com certeza é necessário armazenar os dados do Paciente, os dados e as informações do Médico, os dados para o agendamento da consulta, o Agenda_Consultas e os dados da Consulta (informações do médico durante a consulta).

Assim você identificou grandes grupos de informação que necessariamente precisam ser armazenados para uso futuro na forma de consultas e relatórios (que são processos que atuarão sobre os dados).

A entidade Paciente teria os seguintes atributos.

Nome da entidade: Paciente

Chave identificadora: Nome Paciente

Atributos da entidade:
CPF
Telefone
Endereço
Convênio
Data de nascimento

Exemplos de instâncias do objeto Paciente seriam:

Nome: Paulo Lopes	Nome: Joana da Silva
Telefone: 88890899	Telefone: 8887777
CPF: 483.432.546-65	CPF: 488.444.449-35
Endereço: Rua do Pântano 120	Endereço: Rua do Rio 26

Na figura a seguir, você pode ver um recorte de um diagrama ER. Observe que é a relação entre a entidade paciente que realiza um agendamento na clínica.

Figura 3.10 – Recorte de um diagrama ER
Fonte: Elaboração da autora (2008).

Se você pensar na entidade Paciente existente na clínica, pode representar seus atributos por meio da notação gráfica. O atributo Nome é sublinhado porque é considerado que ele é uma chave nessa entidade. Cada um dos círculos representa um atributo.

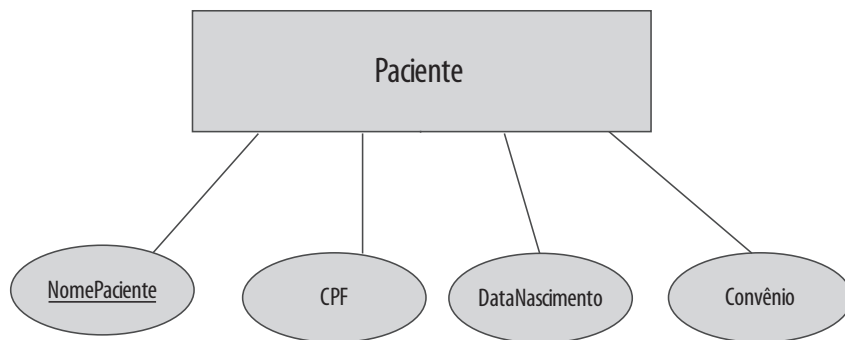



Figura 3.11 – Notação gráfica entidade Paciente
Fonte: Elaboração da autora (2008).

Observe os exemplos a seguir. São exemplos de relacionamento, porém sem a indicação do número de relacionamentos existentes entre eles.

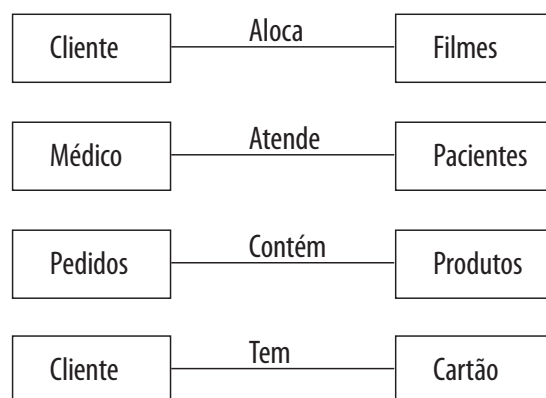



Figura 3.12 – Exemplos de relacionamento
Fonte: Elaboração da autora (2008).

Observe na figura 3.12 as entidades e o relacionamento entre elas:

- **Cliente** (entidade que armazena os dados de um cliente de videolocadora) **Aloca Filme** (entidade que armazena os dados de filmes na locadora).

- **Médico** (entidade que armazena os dados dos médicos na clínica) **Atende Paciente** (entidade que armazena os dados de pacientes na clínica).

- **Pedidos** (entidade que armazena os dados de um pedido de compras) **Contém Produtos** (entidade que armazena os dados dos produtos).

- **Cliente** (entidade que armazena os dados de um cliente de um banco) **Tem Cartão** (entidade que armazena os dados da conta e do cartão que o cliente possui do banco).

Quando você identifica quantas vezes cada instância de uma entidade pode participar do relacionamento, você está determinando a classe desse relacionamento. A cardinalidade indica o número máximo e mínimo de associações possíveis em um relacionamento.

Você pode ter classes de:

a) **Cardinalidade 1 para 1 (1:1) –** Quando a cardinalidade é de 1 para 1, significa que cada instância da primeira entidade pode ser relacionada com exatamente uma instância da segunda entidade. Ou seja, cada cliente tem um cartão, como no exemplo a seguir, em que Ana Lopes tem um, e apenas um cartão, cujo número é 23456-7, e o cartão tem somente 1 cliente, a Ana Lopes.

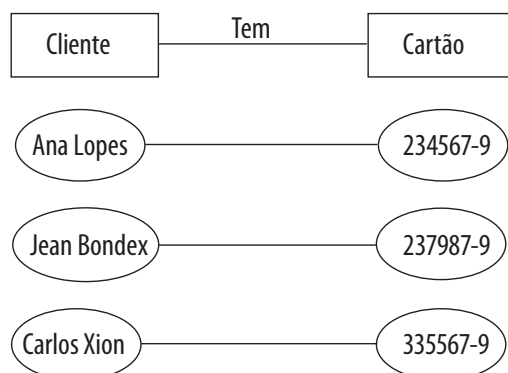



Figura 3.13 – Cardinalidade 1 para 1 (1:1)
Fonte: Elaboração da autora (2008).

b) **Cardinalidade 1 para N (1:N) –** Se a cardinalidade for 1:N, tem-se então uma situação como a ilustração a seguir, na qual um cliente pode ter de zero (nenhuma) a "n" DVDs alocados. A instância João tem três DVDs (instâncias) alocados.

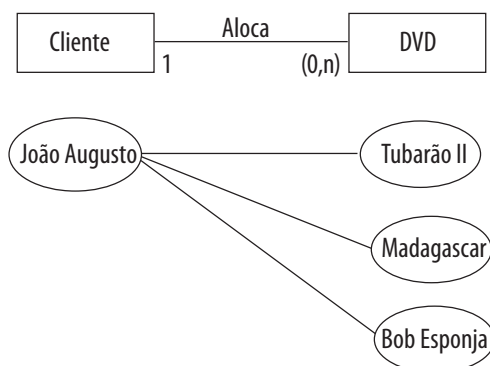



Figura 3.14 – Cardinalidade 1 para N (1:N)
Fonte: Elaboração da autora (2008).

c) **Cardinalidade N para N (N:N) –** A situação N para N representa uma situação de relacionamento de muitos para muitos. Nesse caso, uma instância da primeira entidade se relaciona com uma ou mais instâncias da segunda entidade, que também estabelece relacionamento com uma ou mais instâncias da primeira entidade. No exemplo a seguir, o funcionário João Augusto pode estar relacionado a diversos projetos. O projeto Folha de pagamento tem alocados vários funcionários.

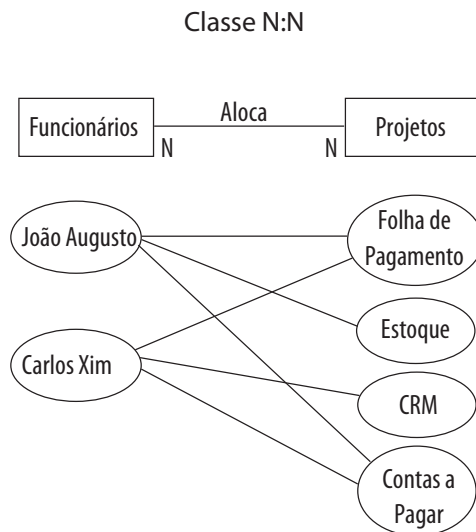



Figura 3.15 – Cardinalidade N para N (N:N)
Fonte: Elaboração da autora (2008).

Observe algumas dicas importantes.

Identifique, primeiro, quais são as entidades do sistema.

- Identifique, então, os atributos para cada entidade.
- Determine as chaves das entidades: lembre-se de que a chave **deve** ser um atributo que distinga a instância de forma única em relação às outras instâncias da entidade: Um exemplo disso é o registro do CPF. Cada cidadão possui o seu, é único, não existe nenhum outro brasileiro que possua o mesmo número. Portanto, o CPF é uma excelente chave.
- Identifique os relacionamentos entre as entidades.
- Por fim, determine a cardinalidade.

Agora pense no caso da Clínica Bem-Estar visto na unidade 2, exercício das atividades de autoavaliação. Vamos identificar pelos menos três entidades:

- Paciente;
- Médico;
- Agenda_Consultas.

Neste caso, o relacionamento seria:

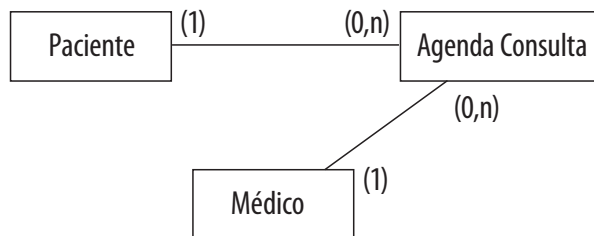



Figura 3.16 – Exemplo de cardinalidade Clínica Bem-Estar
Fonte: Elaboração da autora (2008).

Um paciente pode ter nenhuma, uma ou várias consultas agendadas. Mas para cada consulta só haverá um paciente e haverá relacionado um médico, mas um médico pode ter várias consultas agendadas.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize as atividades propostas a seguir.

Síntese

Nesta unidade, você estudou os principais conceitos adotados nessa metodologia. Durante o estudo, o uso das ferramentas – como o DFD, o dicionário de dados, a descrição do processo e o diagrama ER – foi abordado, evidenciando sua importância na concepção do modelo previsto para a solução do problema do cliente. O uso do DFD torna clara à equipe de projeto e ao usuário a forma como as informações transitarão nos futuros processos e a transformação sofrida durante sua evolução. Foi possível também abordar a importância da normalização, evitando redundâncias na futura base de dados do sistema.

A análise estruturada foi um movimento inicial do uso de uma metodologia de projeto que permitisse documentar o futuro projeto de forma clara e consistente. A evolução, no entanto, era inevitável.

Atividades de autoavaliação

Leia com atenção os enunciados e, após, realize as questões propostas.

1) Relacione os conceitos a seguir, observando que uma mesma opção pode se repetir.

A. Diagrama de fluxo de dados	( )	Notação utilizada para descrever o conteúdo e significado de fluxos, entidades externas, processos e depósitos de dados.
B. Dicionário de dados	( )	Notação que permite representar um processo em um DFD.
C. Descrição dos processos	( )	Utilizado para representar o cliente em um DFD.
D. Fluxo de dados	( )	Utilizado para representar um repositório de dados em um DFD.
E. Fluxo de dados	( )	Permite descrever os fluxos de informação e as transformações sofridas pelos dados durante a execução do processo.
F. Depósito de dados	( )	Utilizado para representar os dados que serão inseridos no processo em um DFD.
G. Processo	( )	Utilizado para representar em um DFD uma empresa que participa de alguma maneira no processo, por exemplo, uma instituição bancária em um processo de contas a receber.

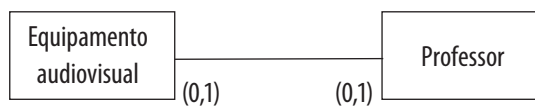
2) Defina a cardinalidade e as entidades existentes para as situações propostas a seguir.

a) Um professor leciona várias disciplinas em sua universidade.

b) A universidade emprega vários funcionários.

c) Os funcionários são lotados em um departamento.

d) Um aluno pode estar matriculado em nenhuma ou várias disciplinas e uma disciplina pode ter vários alunos nela matriculados.

Exemplo de resolução: um equipamento de audiovisual é alocado a um professor.

Saiba mais

Para conhecer um pouco mais sobre a análise estruturada, você deve dar uma olhadinha nos seguintes livros:

DEMARCO, Tom. **Análise estruturada e especificação de sistema**. Rio de Janeiro: Campus, 1989.

YOURDON, Edward. **Análise estruturada moderna**. Rio de Janeiro: Campus, 1992.

UNIDADE 4

4

Visão geral da UML

Objetivos de aprendizagem

- Compreender as diferenças fundamentais existentes entre a análise estruturada e a análise orientada a objetos.
- Perceber as diferentes visões da UML e os diagramas oferecidos para viabilizar seu entendimento.
- Conhecer a história e o surgimento da linguagem de modelagem UML e suas diferentes possibilidades de aplicação.

Seções de estudo

Seção 1 O paradigma da orientação a objetos

Seção 2 A origem das linguagens de modelagem

Seção 3 As cinco visões da UML

Seção 4 Diagramas da UML

Seção 5 Ferramentas

Para início de estudo

Nesta unidade você vai iniciar seu estudo acerca da análise orientada a objetos.

Para compreender esse conceito, é necessário que você perceba suas diferenças em relação à análise estruturada e suas diversas visões relacionadas ao processo de desenvolvimento.

Você pode usar a UML para modelar várias fases de um sistema, desde a análise do problema até a geração do código. A linguagem pode ser aplicada em qualquer tipo de sistema de desenvolvimento de *software*,em sistemas mecânicos de engenharia ou na organização de processos de uma organização.

A UML é hoje uma das abordagens mais utilizadas no mundo, esse sucesso se deve, em parte, por sua padronização, facilidade de reutilização, flexibilidade e possibilidade de abstração de dados.

A partir desta unidade você vai submergir, aos poucos, no mundo orientado a objetos, usando conceitos e ferramentas.

Seção 1 – O paradigma da orientação a objetos

Como já apresentado, a principal ênfase sempre é dada aos procedimentos. Os procedimentos são implementados em blocos e a comunicação entre eles se dá pela passagem de dados. Na orientação a objetos, os dados e procedimentos fazem parte de um só elemento básico. Isso significa que se encontram encapsulados em um só elemento.

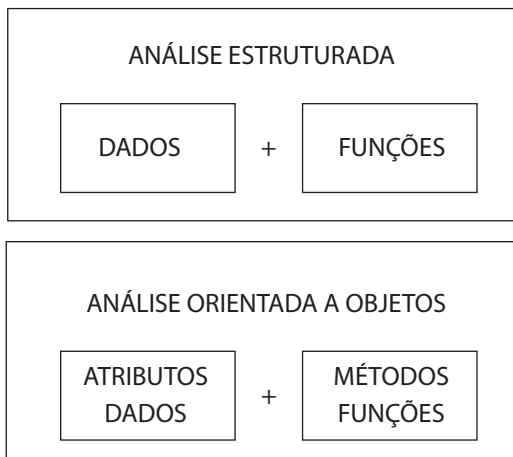



Figura 4.1 – Visão comparativa da análise
Fonte: Pacheco (1994).

Pode-se dizer que, na visão estruturada, a perspectiva adotada é a de um algoritmo, em que o bloco principal de construção do *software* é o procedimento ou a função. A visão do desenvolvedor será a de decompor algoritmos maiores em menores.

Assim, você pode dizer que a análise estruturada desenvolve uma visão de desenvolvimento baseada em um modelo entrada-processamento-saída. Com o passar do tempo, a manutenção pode tornar-se difícil, pois o sistema é totalmente construído a partir do foco do algoritmo.

Quando se adota a visão orientada a objetos, o bloco de construção do *software* passa a ser o objeto ou a classe. Mas você sabe o que é um objeto?

O objeto é uma abstração de conjunto de coisas do mundo real; pode ser uma máquina, uma organização, um carro, uma passagem de ônibus.

A figura 4.2 apresenta "coisas" do mundo real, e portanto são objetos sob o ponto de vista da orientação a objetos.

Figura 4.2 - Objetos
Fonte: Elaboração da autora (2010).

E uma classe? A classe pode ser vista como a descrição de um tipo de objeto, com propriedades semelhantes (atributos), o mesmo comportamento (operações), os mesmos relacionamentos com outros objetos e a mesma semântica. Por exemplo: a classe Aluno de uma escola, com um conjunto de alunos que apresentam as mesmas informações. Todos os objetos são instâncias de classes. A classe, por sua vez, deve descrever as propriedades e os comportamentos daquele objeto.

Uma classe descreve um grupo de objetos.

Observe que os objetos apresentados na figura 4.2 podem ser agrupados por apresentarem atributos, características ou comportamentos semelhantes. Isso permite que sejam criadas as classes Animais, Edificações e Transportes.

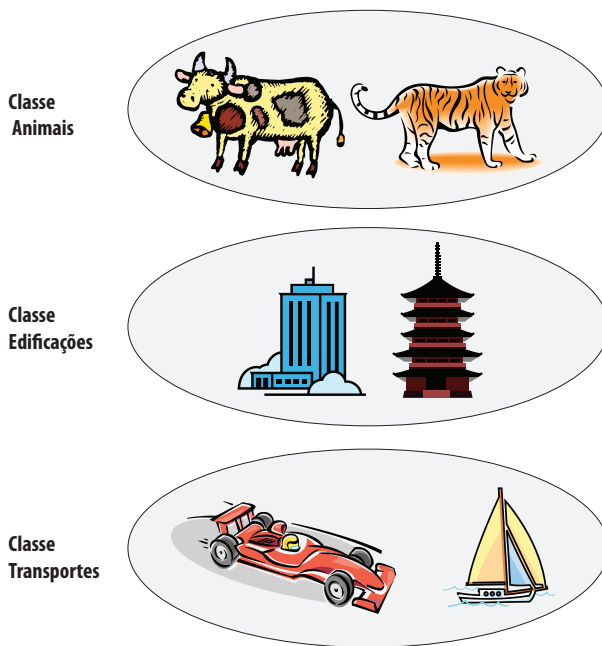



Figura 4.3 – Agrupamento de objetos em classes
Fonte: Elaboração da autora (2010).

Observe que é possível perceber atributos comuns assim como características e comportamentos e, por esse motivo, é possível agrupar esses objetos.

Veja que na classe Animais pode se indicar alguns atributos como espécie, data nascimento, nome que são comuns a qualquer animal, assim como podemos indicar comportamentos comuns como comer, correr, dormir. Essa proximidade é o primeiro passo na identificação de uma classe.

A orientação a objetos pressupõe que o mundo é composto por objetos, sendo um objeto uma entidade que combina estrutura de dados e comportamento funcional. No paradigma orientado a objetos, os sistemas são estruturados a partir dos objetos que existem no domínio do problema, isto é, os sistemas são modelados como um número de objetos que interagem. Mas o que você entende pela expressão "orientado a objetos"?

Se você teve dificuldade em conceituar, não se preocupe. Mesmo *experts* nessa área passaram anos engalfinhando-se, tentando esclarecer seu significado. Pages-Jones (2001), lista seus principais conceitos, que juntos explicam o método:

- **Encapsulamento** – É o agrupamento de ideias afins em uma unidade. Permite ao programador esconder os detalhes da representação dos dados por trás de um conjunto de operações (como a interface). Reflita sobre o seguinte: você sabe como funciona internamente o seu micro-ondas? Mas você sabe como ligá-lo, programá-lo e desligá-lo. Você interage com o micro-ondas por meio de sua interface sem se preocupar com os detalhes da implementação: esse é um exemplo de encapsulamento.

- **Ocultação de informações e implementações** – é exatamente o uso do encapsulamento que restringe a visibilidade de detalhes e informações que ficam internas à estrutura do encapsulamento.

- **Mensagens** – solicitação de um objeto para que outro objeto efetue uma de suas operações. Os objetos mandam mensagens entre si. As mensagens resultam na chamada de métodos que executam as ações necessárias.

- **Classes** – as classes são conjuntos de objetos com características semelhantes.

- **Herança** - o mecanismo de herança permite que uma classe seja criada a partir de outra classe (superclasse), e a nova classe (subclasse) herda todas as suas características.

- **Poliformismo** - é a propriedade segundo a qual uma operação pode se comportar de modos diversos em classes diferentes.

Sistemas orientados a objetos são flexíveis a mudanças, possuem estruturas bem conhecidas e oferecem a oportunidade de criar e implementar componentes totalmente reutilizáveis.

Quer conhecer mais?

Para saber um pouco mais sobre orientação a objetos e suas características, leia o capítulo 1 do livro Fundamentos do desenho orientado a objetos, de Pages-Jones, editora Makron Books, 2001.

Seção 2 – A origem das linguagens de modelagem

As linguagens de modelagem orientada a objetos começaram a surgir na década de 1970. Nessa década, também apareceram no mercado computadores mais modernos e acessíveis, iniciando uma grande expansão computacional.

No início da década de 1990, surgiram as primeiras metodologias orientadas a objeto. Muitos foram os métodos oferecidos como solução para especificação de projetos orientados a objetos, mas alguns se destacaram tornando-se referência por suas características:

- **O método Booch (1993)** – Considerado expressivo nas fases de projeto e construção, apresenta o projeto como um conjunto de visões, cada visão pode ser descrita por modelos e diagramas. A simbologia do modelo é bastante complexa.

- **O método OOSE (*Object Oriented* *Software Engineering*), de Jacobson (1993)** – Excelente no controle da captura de requisitos, análise e projeto de alto nível. Utiliza-se de *use cases* para definir os requisitos iniciais do sistema, vistos por um ator externo.

- **O método OMT (*Object Modeling Technique*) de Rumbaugh (1995)** – Bastante usado para análise e sistemas de informações com uso intensivo de dados. Possui um forte foco para o teste de modelos, baseado nas especificações da análise de requisitos do sistema.

Além dos métodos citados, existiam muitos outros; cada método utilizava-se de notações diferentes. Com o passar do tempo, a percepção de que os métodos poderiam ser complementares a partir de suas melhores características fez com que os três autores se unissem na especificação de um novo método, proporcionando estabilidade e uma linguagem clara e madura que auxiliasse com problemas que, provavelmente, nenhum dos três métodos poderia resolver.

Qual a origem da Linguagem de Modelagem Unificada (UML)?

Em 1994, nas dependências da *Rational Software*, iniciou-se a junção do método Booch e OMT. Em 1995, Jacobson se uniu à equipe e decidiram a incorporação do método OOSE. A primeira versão (0.9) da UML foi lançada ao mercado em 1996. A partir desse lançamento, profissionais da área contribuíram com críticas e sugestões ao modelo. Empresas, como a Hewlett-Packard, I-Logix , DEC, IBM, Microsoft, Oracle Texas, entre outras, investiram maciçamente no aperfeiçoamento do método.

Os pesquisadores Booch, Jacobson e Rumbaugh foram os idealizadores do projeto, mas o produto final em sua versão 1.3 foi resultado de um trabalho de equipe por meio da participação de diversos colaboradores, contribuindo com suas experiências e seu ponto de vista.

Em 1997, a UML foi aprovada como padrão pela *Object Management Group (*OMG).

A UML é uma linguagem completamente visual. Por meio de elementos gráficos ela permite a representação de conceitos da orientação a objetos utilizados na estrutura de elaboração de um projeto de *software*.

Você pode representar o sistema por diferentes perspectivas dependendo do diagrama que você utilizará. Cada um de seus diagramas possui uma forma predeterminada de desenhar o elemento, uma semântica que define o significado desse elemento e onde ele pode ser utilizado.

Não existe dependência de linguagem no uso da UML ou de processos de desenvolvimento. Ela pode servir para qualquer linguagem que o desenvolvedor venha a utilizar.

Além de ser utilizada como uma linguagem de especificação para construir modelos, a UML oferece a possibilidade de conectar seus modelos a diversas linguagens como JAVA, C++, Visual Basic e, inclusive, bancos de dados. Em outras palavras, você pode gerar código a partir de um modelo UML em uma linguagem de alto nível como JAVA.

Seção 3 – As cinco visões da UML

Na UML todas as abstrações de um sistema são organizadas em modelos, no qual cada um deles representa uma visão do sistema.

Se você olhar para o desenvolvimento como um todo, perceberá etapas bem definidas. Cada etapa pode ser representada por meio de diagramas e modelos de elementos, de forma a proporcionar ao projetista uma visão eficiente e completa daquela etapa. Então, você pode assumir que as visões mostram diferentes aspectos do sistema que está sendo modelado.

A visão vai apresentar um aspecto particular do sistema.

Segundo Bezerra (2002), a UML pode ser apresentada por cinco visões:

- **Visão de casos de uso** – Descreve a funcionalidade do sistema desempenhada pelos usuários. A visão *use case* é central, pois seu conteúdo é fundamental para a composição das demais visões do sistema.

- **Visão de projeto** – São enfatizadas as características do sistema que dão suporte estrutural e comportamental.

- **Visão de implementação** – Abrange o gerenciamento de versões do sistema construídas por meio do agrupamento de módulos e subsistemas.

- **Visão de implantação** – Corresponde à distribuição física do sistema em seus subsistemas e a conexão entre essas partes.

- **Visão de processo** – Esta visão enfatiza as características da concorrência, sincronização e desempenho do sistema.

O uso das visões durante o projeto depende do tipo de projeto a ser construído. A visão do processo, por exemplo, é irrelevante se o sistema for construído sobre apenas um processo. Se o sistema é composto por apenas um módulo, é redundante a utilização da visão de implementação.

A empresa decide pela escolha das visões que serão contempladas e, consequentemente, os diagramas, que serão desenvolvidos no projeto a partir do tipo de domínio do projeto, modelo de desenvolvimento, riscos, restrições e características do processo.

Seção 4 – Diagramas da UML

Um diagrama procura representar graficamente a projeção de um sistema. Alguns elementos aparecem em mais de um diagrama, alguns em todos, outros em nenhum.

A UML possui nove diagramas (Furlan, 1998):

a) **Diagrama de casos de uso** – descrevem a funcionalidade do sistema percebida por atores externos. O ator (um usuário, um equipamento, uma organização) interage com o sistema.

b) **Diagrama de classes** – representa a estrutura estática do sistema, as classes, os relacionamentos entre suas instâncias (objetos), restrições e hierarquias.

c) **Diagramas de interação** – é formado por:

- Diagramas de sequência: mostram a colaboração dinâmica entre um número de objetos. O objetivo principal é mostrar a sequência de mensagens enviadas entre objetos.

- Diagramas de comunicação: têm o mesmo propósito dos diagramas de sequência. São desenhados como diagramas de objetos, onde são mostradas as mensagens trocadas entre os objetos.

d) **Diagramas de estados** – mostram as sequências de estados do objeto a partir dos estímulos recebidos, suas respostas e ações. É uma variação dos diagramas de

estado, em que a maioria dos estados é estado de ação e a maioria das transições é ativada por conclusões das ações.

e) **Diagramas de implementação** – é composto por dois diagramas:

- Diagrama de componentes: são mostradas as dependências entre componentes de *software* (código-fonte, código binário e componentes executáveis).

- Diagrama de implantação: mostra elementos de configuração do processamento em tempo de execução, os componentes de *software*, processos e dispositivos físicos.

Os diagramas da UML podem ser situados a partir de suas cinco visões:

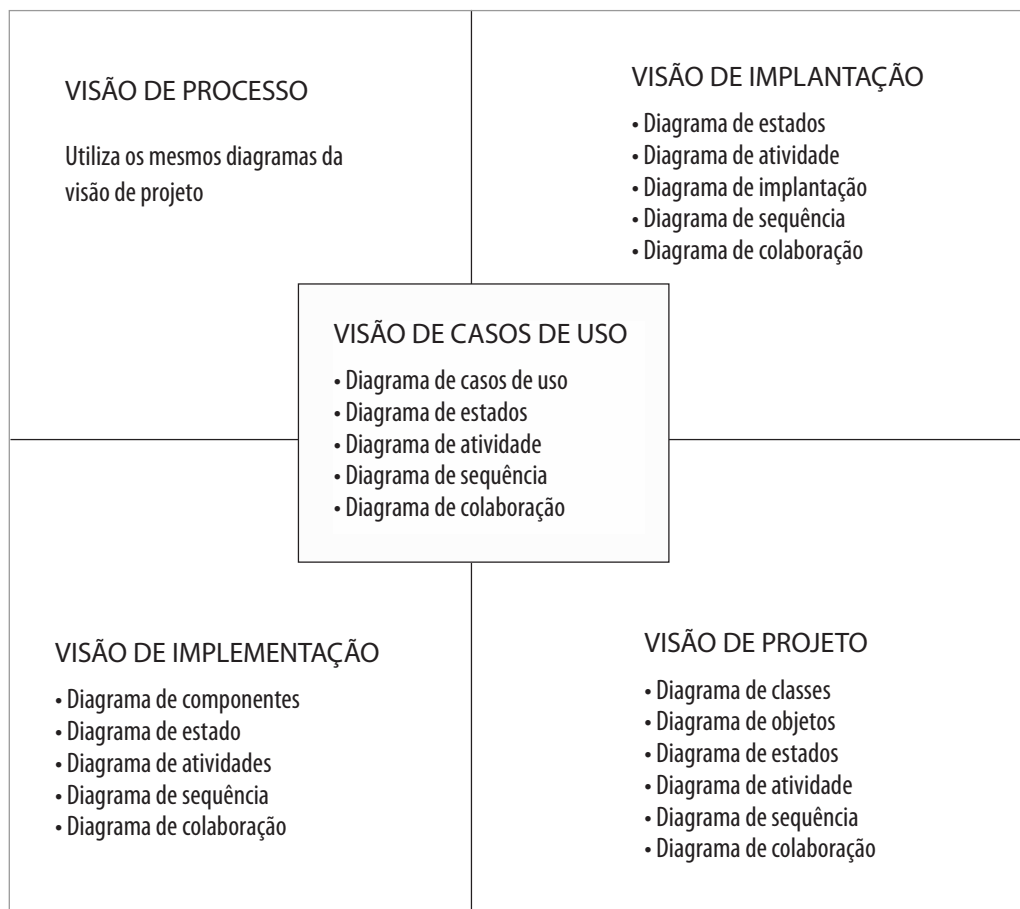



Figura 4.2 – As 5 visões dos diagramas da UML
Fonte: Booch (2000).

Os aspectos estáticos do sistema são capturados pelos diagramas de classes, de objetos e de componentes; os aspectos dinâmicos, pelos diagramas de casos de uso, de estado, de atividade, de seqüência e de colaboração. O modelo funcional é suportado pelos diagramas de componente e execução.

Seção 5 – Ferramentas

Para modelar um sistema utilizando a notação UML, é fundamental que você utilize uma **ferramenta** que automatize o método.

A escolha de alguma dessas ferramentas é fundamental para que você continue seus estudos.

Como a UML utiliza-se de uma notação gráfica, uma boa ferramenta agiliza o processo de construção e recuperação da informação.

As ferramentas disponíveis subdividem-se na categoria *software* livre e demos (ferramentas pagas que oferecem 30 dias para avaliação).

A seguir estão listadas algumas delas:

a) *Software* livre:

- **Orquídea** – É uma ferramenta case, que possui as seguintes funcionalidades: construção de diagrama de classes e de diagramas de sequência na notação UML, geração e leitura de código C++; geração de documentação web em HTML ou HTM.

- **EclipseUML** – É uma plataforma aberta para integração de ferramentas criada por uma comunidade aberta de provedores de ferramentas. A IDE Eclipse foi criada sobre o paradigma *Open Source* baseada na *Common Public License*.

- ***Omondo* EclipseUM*L*** – É um *plugin* para a Eclipse que auxilia na construção de diagramas UML. Com esse *plugin*, você pode criar diagramas de classe, sequência, estados, *use cases*, atividades etc. Alterações no diagrama automaticamente se refletem no código-fonte e vice-versa.

b) Versões demo:

- ***Rational Rose*** – Ferramenta de modelagem que suporta todos os diagramas previstos na linguagem de modelos UML. Seu custo é extremamente alto, sua grande vantagem é que ela pertence a Rational, originalmente criadora da linguagem UML.

- ***Enterprise Architect* (EA)** – Ferramenta da Sparxsystems da Austrália, suporta todos os diagramas previstos pela UML e toda notação da OMG. Gera códigos e engenharia reversa para manutenção dos modelos. Apresenta uma interface mais amigável e simples de se usar.

- **Poseidon** – Baseada no mesmo *engine* do ArgoUML, mas com melhorias. Possui uma *Comunity Edition* gratuita. Não permite funcionalidades de impressão de diagramas, mas possibilita a exportação das imagens dos diagramas.

A partir da próxima unidade, você vai iniciar a confecção de diagramas, sendo necessário o uso de uma ferramenta. Todas as **ferramentas** apresentam, nas respectivas páginas de *download*, tutoriais e manuais de instalação e utilização.

Na midiateca você também encontra documentação sobre algumas das ferramentas.

Quer conhecer mais?

Se você quer aprofundar seu conhecimento sobre as diferentes visões da UML, leia o capítulo 2, *Introdução a UML*, do livro **UML: guia do usuário**, de BOOCH, RUMBAUGH e JACOBSON, editora Campus, 2000.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize as atividades propostas a seguir.

Síntese

Você estudou as diferenças existentes entre a análise estruturada e a análise orientada a objetos. Você percebeu que a UML surgiu a partir da evolução de outros métodos orientados a objetos e que, de certa maneira, complementaram-se dentro de uma linguagem de modelagem padronizada.

Ao estudar a unidade, apresentaram-se as diferentes visões que, de diferentes formas, apoiam os projetistas na melhor visualização de todas as etapas do ciclo de desenvolvimento do *software*. Além da introdução conceitual, da análise orientada a objetos e à UML, também foram apresentadas algumas ferramentas que suportam o método.

Atividades de autoavaliação

Leia com atenção os enunciados e, após, realize as questões propostas.

1) É correto afirmar que (pode haver mais de uma afirmativa correta):

a) ( ) A análise estruturada possui como fundamento a visão da funcionalidade. Assim, a visão que se impõe ao modelo é o processamento dos dados.

b) ( ) Na análise orientada a objetos, dados e métodos são vistos como uma entidade única, preocupando-se com propriedades e comportamentos do objeto.

c) ( ) A análise estruturada possui como fundamento a visão do comportamento do processo. Assim, a visão que se impõe ao modelo é o processamento dos dados.

d) ( ) A análise orientada a objetos é construída a partir de objetos. As classes são instâncias do objeto.

2) Identifique os elementos a seguir com C para **c**lasse e O para objetos.

a) ( ) Caixa

b) ( ) Imposto pago

c) ( ) João da Silva

d) ( ) Valor Venda

e) ( ) Cliente

3) Observe os conceitos a seguir e realize as devidas inserções.

Encapsulamento **Herança** **Poliformismo** **Mensagens**

a) ___________ pode explicar situações nas quais pode haver várias formas de fazer uma determinada "coisa".

b) ___________ permite que detalhes internos sejam "escondidos".

c) ___________ especifica informações a serem passadas para a operação que deve ser executada por um objeto receptor.

d) Você define uma classe chamada Conta (para um sistema bancário) com características e comportamento genérico. Posteriormente, você define duas classes, chamadas Poupança e Conta_Corrente; cada uma delas possui propriedades específicas que a outra não possui, mas agregam a elas o comportamento genérico da classe Conta. Isso é chamado de ___________.

4) As visões da UML permitem o uso de diferentes diagramas, adaptando o uso do diagrama às necessidades do projeto. Relacione conceitos e diagramas utilizados ao tipo de visão associado:

A. Visão de casos de uso

B. Visão de projeto

C. Visão de implementação

D. Visão de implantação

E. Visão de processo

a) ( ) Descreve a funcionalidade do sistema desempenhada pelos usuários.

b) ( ) A visão é representada por meio dos diagramas de classe, de objetos, de estados, de atividade, de sequência e de colaboração.

c) ( ) Corresponde à distribuição física do sistema em seus subsistemas e a conexão entre essas partes.

d) ( ) Abrange o gerenciamento de versões do sistema, construídas por meio do agrupamento de módulos e subsistemas.

e) ( ) A visão é representada por meio dos diagramas de casos de uso, de estados, de atividade, de sequência e de colaboração.

f) ( ) Enfatiza as características da concorrência, sincronização e desempenho do sistema.

g) ( ) São enfatizadas as características do sistema que dão suporte estrutural e comportamental.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar em:

GILLEANES, T. A. Guedes. **UML**: uma abordagem prática. São Paulo: Novatec, 2004.

UNIDADE 5

5

Modelagem de casos de uso

Objetivos de aprendizagem

- Compreender a importância da utilização de casos de uso para identificação clara dos objetivos do usuário.
- Entender o significado dos diferentes elementos existentes em um caso de uso.
- Reconhecer meios para identificar atores e casos de uso.
- Compreender o mecanismo de documentação e sua importância na descrição dos casos de uso.

Seções de estudo

Para início de estudo

Nesta unidade, você vai iniciar o estudo sobre os diagramas oferecidos na UML e suas representações utilizadas na modelagem orientada a objetos.

Este estudo inicia com os casos de uso que representam o eixo central do modelo, pois descrevem a sequência de interações realizadas pelo sistema que visam prover algum valor mensurável ao usuário do sistema. O caso de uso permite mapear o escopo do sistema facilitando a comunicação com o usuário do sistema e facilitando a gerência do projeto.

Mas, para descrever o caso de uso, não é suficiente concebermos apenas o diagrama de caso de uso, é preciso entender todos os seus elementos e sua documentação.

Nesta unidade, você vai iniciar seu processo de aprendizagem identificando e construindo os casos de uso necessários para descrever essa visão do projeto.

Seção 1 – O que são casos de uso?

Um sistema sempre interage com usuários, outros sistemas ou equipamentos. Todos eles esperam pelos resultados dessa interação que normalmente são previsíveis ou pelo menos deveriam ser.

Um caso de uso procura documentar as ações necessárias, os comportamentos e as sequências para que o resultado esperado pelo usuário ocorra. Então você pode dizer que modelo de casos de uso modela os requisitos funcionais do sistema.

Antes de iniciar a definição dos casos de uso, você deve estar consciente dos requisitos funcionais e não funcionais necessários ao sistema

Como o caso de uso não especifica detalhes de implementação, você vai, na verdade, pensar em como o sistema será utilizado. Agora imagine a seguinte situação: você realizará o projeto usando UML para uma pequena videolocadora. Nesse pequeno projeto, os requisitos funcionais listados são:

- O gerente deseja cadastrar seu acervo de DVDs com informações gerais sobre cada filme, como: nome do filme; duração; diretor; principais atores; gênero; idiomas disponíveis.

- É fundamental que o atendente possa realizar o cadastro de clientes possibilitando o cadastro de endereço, telefone do cliente e três possibilidades para digitação de nomes de pessoas autorizadas para retirar DVDs.

- É necessário que o sistema permita o controle de entrega e recebimento dos DVDs.

- O gerente deseja um relatório estatístico dos DVDs mais locados

- Deve ser possível um relatório de consulta dos DVDs em atraso.

- Relatório que apresente os 50 maiores clientes da locadora.

Agora tente identificar os casos de uso. Então, quantos você encontrou?

Você pode listar os seguintes casos de uso:

- **Gerenciar cliente** – será responsável por cadastro, consulta, exclusões e alterações de dados cadastrais do cliente.

- **Gerenciar filmes** – será responsável por cadastro, consulta, exclusões e alterações de dados dos DVDs da videolocadora.

- **Gerenciar locações** – deve permitir o controle de entregas e recebimentos dos DVDs e cálculo de multas, quando necessário.
- **Gerenciar relatórios** – deve possibilitar a impressão dos relatórios estatísticos de locação, maiores clientes e controles de atraso de entrega.

Se você representar graficamente os casos de uso, terá os mesmos representados por um círculo. Veja a figura 5.1.

Retome o exemplo da videolocadora. A figura 5.1 mostra três casos de uso possíveis para esse projeto.

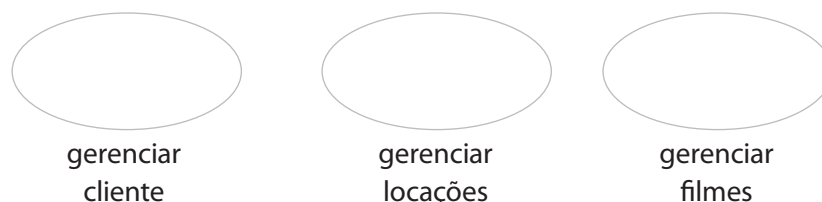



Figura 5.1 – Casos de uso de uma videolocadora
Fonte: Elaboração da autora (2008).

O modelo de casos de uso é composto por casos de uso, atores e relacionamentos.

O caso de uso descreve um conjunto de sequências, cada um representando a interação entre atores com o próprio sistema. Esses comportamentos são funções em alto nível que você vai visualizar, especificar, construir e documentar tentando mostrar de forma clara o comportamento pretendido do sistema durante a análise de requisitos.

Quando você nomear o caso de uso, esse nome deve ser único dentro do projeto, diferenciando-o dos demais casos de uso. Você pode usar qualquer caractere textual no nome do caso de uso, mas evite o uso de dois-pontos. Procure utilizar expressões verbais ativas.

O que é o diagrama de casos de uso?

Se um caso de uso é sempre iniciado a partir do momento em que um ator envia sua mensagem (estímulo), o diagrama de casos de uso é criado para visualizar os relacionamentos entre atores e casos de uso.

O diagrama de casos de uso pode ser utilizado para representar um caso de uso e seus relacionamentos, todos os casos de uso para um ator ou ainda todos os casos de uso a serem implementados em um ciclo de desenvolvimento.

A notação do diagrama faz uso de um boneco para identificar o ator, abaixo do boneco é inserido o nome do ator cliente. A elipse identifica o caso de uso, o nome do caso de uso pode ser escrito no interior ou externo a elipse (caso de uso: gerenciar cliente). A linha reta indica o relacionamento de comunicação entre o ator e o caso de uso.

Figura 5.2 – Caso de uso gerenciar cliente
Fonte: Elaboração da autora (2008).

Seção 2 – Como identificar os atores?

Um ator representa um conjunto coerente de papéis que os usuários de casos de uso desempenham quando interagem com esses casos de uso. Um ator pode representar um papel que um ser humano, um dispositivo de *hardware* ou até outro sistema que desempenha alguma interação com o sistema (BOOCH, 2000).

O ator troca informações com o sistema, ele não faz parte do sistema apenas interage com ele. Um ator pode participar de vários casos de uso.

Você pode ter em seu sistema diferentes tipos de atores:

- pessoas (professor, aluno, secretária, coordenador);
- organizações (empresa fornecedora, bancos, receita federal);
- outros sistemas (módulos que venham a interagir com seu sistema, como contas a pagar, crediário, estoque); e
- equipamentos (coletor de dados, leitora de código de barras, balanças).

Quando você escolhe o nome do ator lembre-se de que ele representa um papel no sistema, nunca utilize nomes pessoais. Imagine: um indivíduo pode representar o papel de funcionário em alguns momentos e em outros pode representar o papel de gerente.

Nesse caso, nomes que representam o papel poderiam ser Gerente, Professor, Vendedor, Fornecedor.

Mas como identificar os atores do sistema?

Primeiro você deve identificar quais as fontes de informação que serão processadas e quais os destinos das informações geradas.

Sempre avalie quais os departamentos da empresa que serão afetados e que potencialmente podem ter atores que interagem com o sistema.

Bezerra (2002) oferece algumas dicas na forma de perguntas que apoiam a descoberta dos atores do sistema:

- Que órgãos, empresas ou pessoas utilizarão o sistema?
- Que outros sistemas irão se comunicar com o sistema a ser construído?
- Alguém deve ser informado de alguma ocorrência no sistema?
- Quem está interessado em um certo requisito funcional do sistema?

Figura 5.3 – Exemplos de atores
Fonte: Elaboração da autora (2008).

Você consegue listar os atores da videolocadora? Pense um pouco sobre o assunto.

Os possíveis atores são:

- o atendente;
- o gerente;
- o cliente.

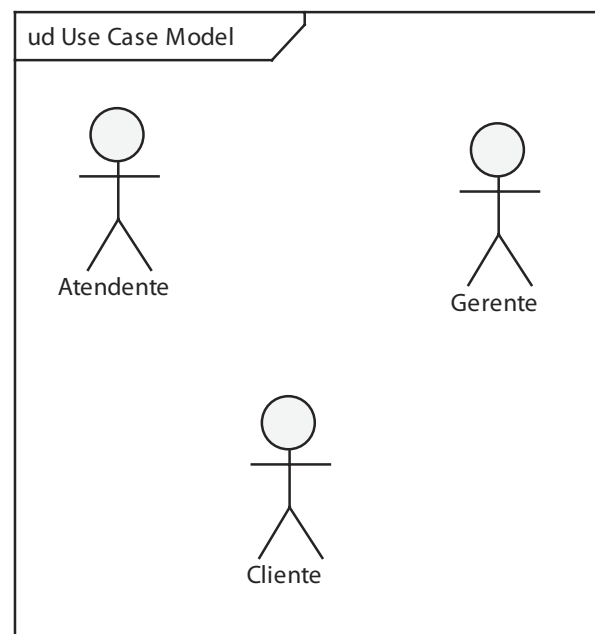



Figura 5.4 – Diagrama de atores da videolocadora
Fonte: Elaboração da autora (2008).

Só definir os atores não é o suficiente, você deve descrever características e responsabilidades desses atores. Características importantes podem ser cargos, funções, grau de escolaridade, permissões de acesso, frequência de uso, conhecimento em informática, conhecimento no processo do negócio.

Os atores podem ser divididos em primários e secundários, o ator primário inicia a sequência de ações de um caso de uso. Já os atores secundários supervisionam, operam, mantêm ou auxiliam na utilização do sistema.

Nro.	Ator	Descrição
1	Cliente	• Definição – indivíduo que realiza locações de DVDs na videolocadora. • Frequência de uso – diário, semanal • Conhecimento em informática – relativo, alguns clientes possuem outros não • Conhecimento no processo – sim, à grande maioria possui uma noção clara do funcionamento do processo de locação de DVDs • Grau de escolaridade – desde fundamental a pós-graduação • Permissões de acesso – deve ser disponibilizado ao cliente a consulta ao acervo.
2	Gerente	• Definição – funcionário da videolocadora responsável por operações de abertura, fechamento, controle de funcionário, controle de compras e pagamentos da videolocadora. • Frequência de uso – diário • Conhecimento em informática – aplicativos *Word, Browsers, Windows* XP • Conhecimento no processo – domina todo o processo do negócio • Grau de escolaridade – graduação • Permissões de acesso – terá acesso a todas as funcionalidades do sistema

Quadro 5.1 - Exemplo de descrição de atores
Fonte: Elaboração da autora (2008).

A descrição detalhada do ator ajuda na definição de perfis que influenciam no projeto da interface.

Seção 3 – Relacionamentos entre casos de uso e atores

Para que um caso de uso seja executado, é necessária a existência de atores que interajam com o caso de uso. Essa interação ocorre por meio dos relacionamentos de comunicação, inclusão, extensão e generalização.

Quando se usa um relacionamento de comunicação, você estará representando quais atores estão associados a um caso de uso. Isso significa que o ator vai interagir trocando informações com o caso de uso. É o relacionamento mais comum onde ocorre troca de informações.

Figura 5.5 – Exemplo de relacionamento de comunicação
Fonte: Elaboração da autora (2008).

O ator cliente oferece informações ao caso de uso como nome e título do filme que deseja, já o ator atendente oferece ao caso informações como o código.

Agora observe o diagrama geral de casos de uso a partir dos requisitos funcionais apontados na seção 1.

O diagrama geral apresenta de forma genérica os casos de uso solicitados sendo que todo o relacionamento existente é de comunicação.

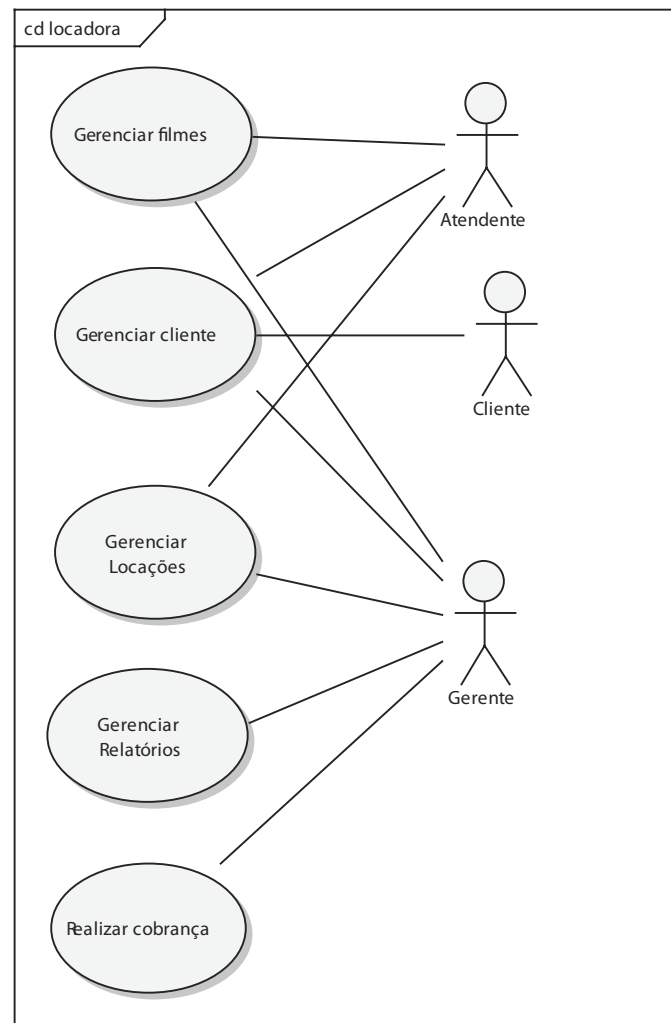



Figura 5.6 – Diagrama de casos de uso da videolocadora
Fonte: Elaboração da autora (2008).

O que é relacionamento de extensão?

Os casos de uso possuem na maioria das vezes um fluxo principal e vários fluxos alternativos. Os casos de uso secundários são muitas vezes inseridos no caso de uso por meio do relacionamento de extensão.

O relacionamento de extensão deve ser usado para representar:

- um comportamento opcional;

- um comportamento que só ocorre sob certas condições (alarmes, por exemplo);
- em fluxos alternativos dependentes da escolha de um ator.

Para entender melhor, considere a seguinte situação:

Quando o ator decide executar o caso de uso extensor, ele executa e após sua execução retorna ao caso de uso estendido.

Você pode comparar um caso de uso extensor a uma função que você desenvolve em um algoritmo de programação.

Veja a seguinte situação:

Em nosso estudo de caso da videolocadora, o contratante do projeto deseja a inserção no cadastro do cliente das preferências do cliente, por exemplo, as preferências do cliente a respeito do tipo de filme (comédia, romance, ficção etc.), diretores, atores.

Para você inserir esta funcionalidade na modelagem dos casos de uso segue o seguinte raciocínio:

- O caso de uso gerenciar cliente vai descrever a sequência de interações para inserir os dados cadastrais do cliente.
- Normalmente o cliente informa apenas as informações cadastrais e salva seus dados, esse é então o fluxo normal de interações.
- MAS caso o cliente deseje, ele pode ainda inserir informações sobre as suas preferências relacionadas aos filmes.
- Esta possibilidade de inserir preferências é opcional e depende da solicitação do usuário, então pode ser definido como um caso de uso estendido.

Veja como esse relacionamento é representado :

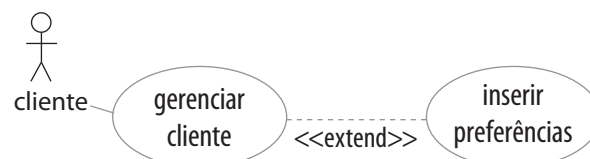



Figura 5.7 – Relacionamento de extensão
Fonte: Elaboração da autora (2008).

O diagrama geral de casos de uso atualizado seria então:

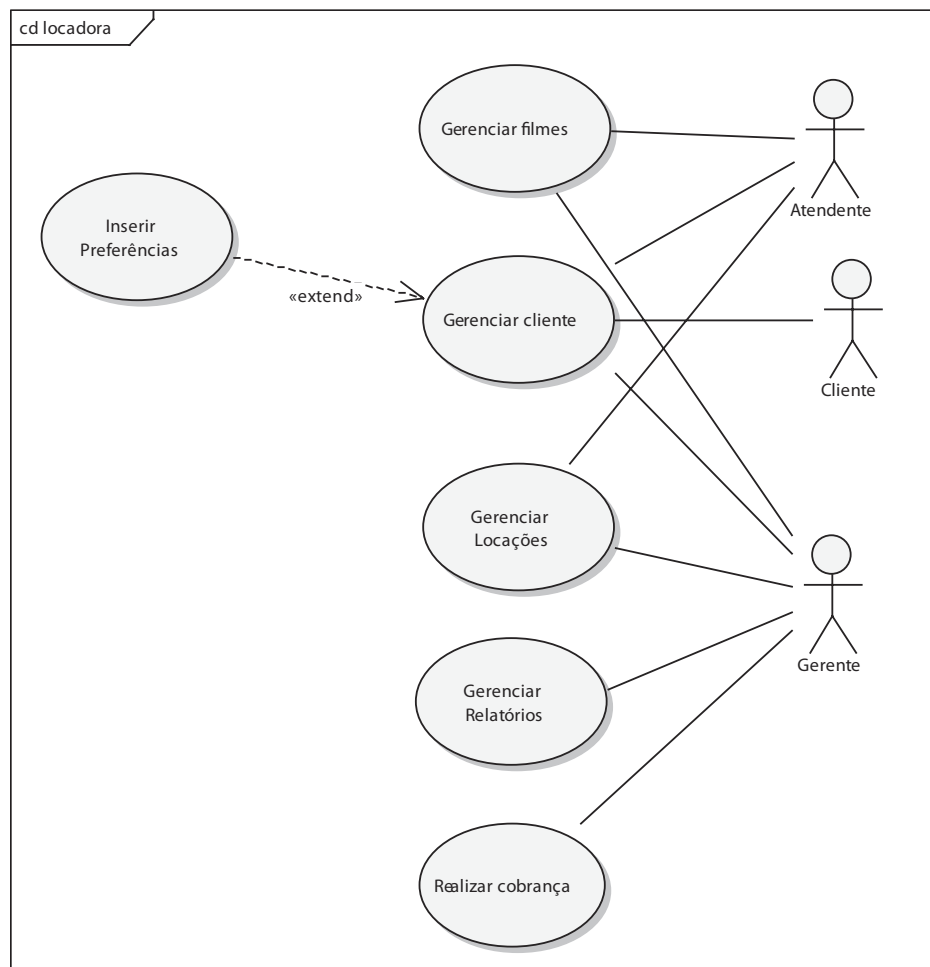



Figura 5.8 – Diagrama de casos de uso da videolocadora
Fonte: Elaboração da autora (2008).

Utilize a extensão somente quando um comportamento opcional de um caso de uso precisar ser descrito.

Quando você faz uso da generalização significa que um caso de uso ou um ator herda características de um caso de uso ou um ator mais genérico.

Nesses casos existem características comuns que podem ser compartilhadas, um relacionamento entre um elemento geral e um outro mais específico. Nesse caso, o elemento mais específico possui todas as características do elemento geral e além destas contém ainda mais particularidades.

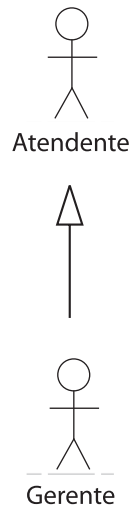



Figura 5.9 — Herança entre atores
Fonte: Elaboração da autora (2008).

Na generalização entre atores, como na figura acima, o ator gerente herda todos os casos de uso do ator atendente.

O ator atendente pode realizar os casos de uso gerenciar filmes, gerenciar cliente e gerenciar locações . O ator gerente herda todos esses casos de uso (pode realizar gerenciar filmes, cliente e locações), mas além destes, o ator gerente pode realizar o caso de uso gerenciar relatórios (que são específicos para esse ator).

Se a generalização for entre casos de uso, então podemos ter uma situação como a da figura 5.10.

Imagine uma funcionalidade no sistema da videolocadora onde será controlada a cobrança de clientes que sejam devedores, a cobrança pode ser realizada pessoalmente ou por telefone.

Nessa situação, o caso de uso Pessoalmente e Telefone, herda o comportamento do caso de uso realizar Cobrança, ou seja, além de herdar comportamentos básicos do caso realizar Cobrança, os dois casos possuem também fluxos de interações específicas para cada um deles.

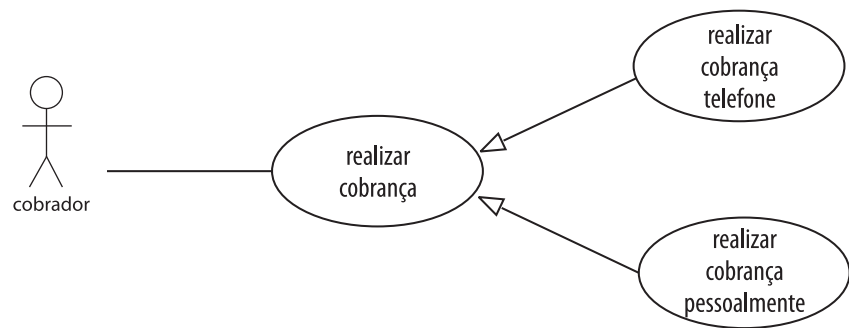



Figura 5.10 – Herança entre casos de uso
Fonte: Elaboração da autora (2008).

O diagrama geral de casos de uso atualizado seria então:

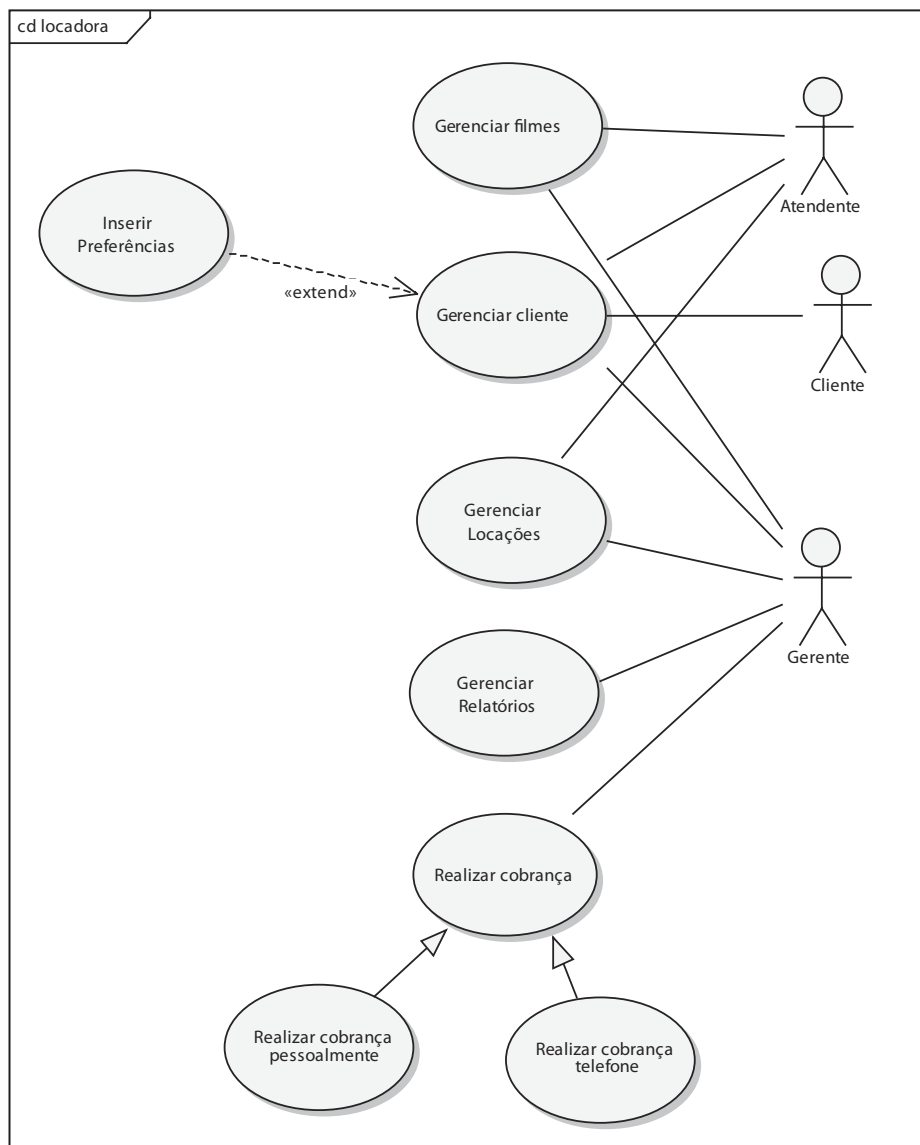



Figura 5.11 – Diagrama de casos de uso de videolocadora
Fonte: Elaboração da autora (2008).

O que é relacionamento de inclusão?

O caso de uso A inclui o caso de uso B quando representa uma atividade complexa e comum a vários casos de uso (PÁDUA, 2001). Esse relacionamento só é possível entre casos de uso.

Em outras palavras, o caso de uso pode ser incluído quando a sequência de interações dele ocorre em mais de um caso de uso.

Ainda pensando no sistema da videolocadora observa-se que apenas o gerente tem acesso a cobrança e aos relatórios do futuro sistema. Para obter o acesso seguro, o sistema deve apresentar uma funcionalidade que permita o controle do acesso por meio da autenticação do usuário no sistema. Essa autenticação será realizada para todos os casos de uso, verificando se o acesso está sendo realizado por um usuário credenciado a usar a função no sistema.

Nessa situação, todos os casos de uso farão uso do caso de uso Autenticação, sendo um caso típico de inclusão, pois o mesmo caso de uso está relacionado a vários casos de uso.

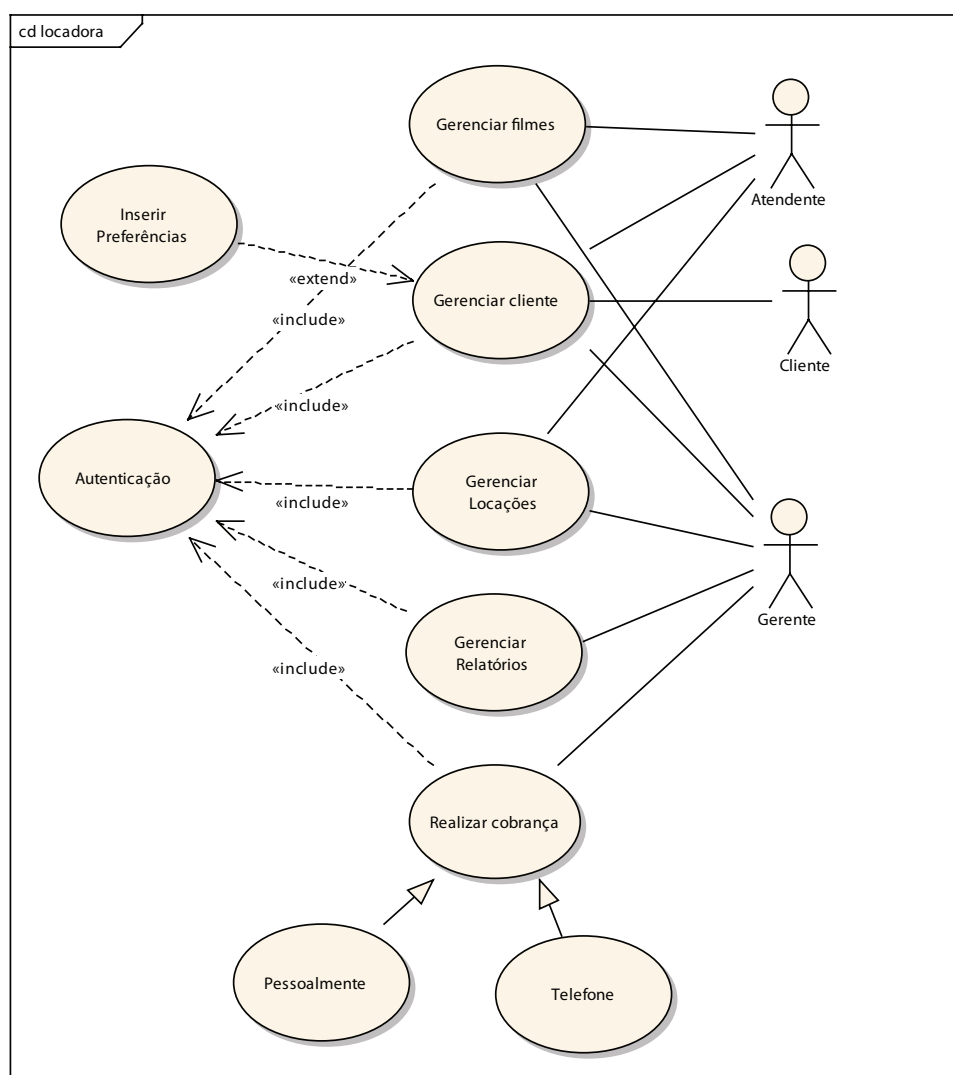



Figura 5.12 – Diagrama de casos de uso da videolocadora
Fonte: Elaboração da autora (2008).

Um exemplo comum de inclusão é a autenticação por meio de contas e senhas. Quando você está em um caixa eletrônico com as possibilidades de saque, extrato, pagamento, apesar de cada uma delas possuir sequências de interações e comportamento específico, têm em comum a autenticação de conta e senha. Logo, esse caso pode ser incluído, pois em todos os processos ele vai acontecer com a mesma sequência de interações.

Utilize o relacionamento de inclusão em situações onde o comportamento se repete em mais de um caso de uso.

Seção 4 – Identificando casos de uso

Mas, como identificar os casos de uso do sistema?

Identifique os objetivos do usuário e não funções no sistema. Para identificar esses casos de uso, Pádua (2001) sugere:

- Verifique quais as tarefas de cada ator.
- Que informação cada ator cria, armazena, consulta, altera ou remove.
- Que informação cada caso de uso cria, armazena, consulta, altera ou remove.
- Que mudanças externas súbitas devem ser informadas ao produto pelos atores.
- Que ocorrências no produto devem ser informadas a algum ator.

Imagine uma situação em que você é convidado a desenvolver um projeto para um caixa eletrônico bancário. O projeto prevê o atendimento dos seguintes requisitos funcionais:

- O sistema deve permitir ao cliente a emissão de saldo somente da conta corrente.
- O sistema deve permitir ao cliente a emissão de extrato somente da conta corrente.
- O sistema deve permitir a atualização dos dados cadastrais do cliente.
- O sistema deve permitir o saque em dinheiro no caixa eletrônico.

- O sistema deve permitir a consulta a toda a movimentação financeira do cliente (conta corrente, poupança e aplicações) no caixa eletrônico.
- O acesso as funcionalidades do sistema deve ser possível somente após a verificação da conta e senha do cliente ou gerente.

Figura 5.13 – Diagrama de casos de uso do caixa eletrônico
Fonte: Elaboração da autora (2008).

Como documentar o caso de uso?

Ao definir um caso de uso, é necessária uma descrição narrativa das interações entre os elementos externos e o sistema.

E o que deve ser documentado para o caso de uso? Existem alguns aspectos que são fundamentais:

Campo	Descrição
Nome	O nome do caso de uso deve ser o mesmo nome que consta no diagrama. Não esqueça de que o nome de um caso de uso deve ser único.
Identificador	Convenção numérica utilizada para identificar o caso de uso. Exemplo: CSU002, CSU001.
Descrição	Descrição sucinta do caso de uso.
Ator Primário	O nome do ator que é o responsável pelo início do caso de uso.
Ator Secundário	Outros atores que fazem parte do caso de uso.
Precondição	A precondição indica as condições necessários no sistema para que o caso de uso ocorra. Ex: Para realizar o caso de uso saldo, a precondição pode ser a validação positiva de conta e senha.
Fluxo Principal	O fluxo principal descreve a sequência de ações que deve ocorrer quando o caso de uso é realizado.Seja breve na descrição, o fluxo deve ser escrito utilizando-se a terminologia do usuário.
Fluxo Alternativo	Descreve a sequência de ações quando o ator faz uma escolha alternativa. O fluxo alternativo descreve um comportamento alternativo para o fluxo principal.
Pós-condição	Você deve descrever aqui o estado do sistema após a execução do caso de uso.
Regras de Negócio	Condições ou restrições na execução do caso de uso.

Quadro 5.2 - Documentação do caso de uso
Fonte: Bezerra (2000).

A seguir, um exemplo do diagrama de caso de uso documentado:

Nome	Realizar saldo
Identificador	CSU001
Descrição	O cliente do banco solicita a emissão do saldo de sua conta corrente.
Ator Primário	Cliente
Precondição	Conta e senha do cliente serem válidas.
Fluxo Principal	1. o cliente informa a conta e agencia 2. o cliente informa a senha 3. o sistema exibe a lista de serviços do caixa eletrônico 4. o cliente seleciona a opção saldo 5. o sistema apresenta mensagem informando a emissão do saldo
Fluxo Alternativo	1. O cliente cancela a operação por teclado finalizando o caso de uso
Pós-condição	Saldo do cliente impresso
Regras de negócio	RN01

Quadro 5.3 - Exemplo de documentação do caso de uso realizar saldo
Fonte: Elaboração da autora (2008).

Nome	Realizar saque
Identificador	CSU003
Descrição	O cliente do banco solicita saque de sua conta corrente.
Ator Primário	Cliente
Precondição	Conta e senha do cliente serem válidas. Haver saldo na conta superior ou igual ao valor solicitado.
Fluxo Principal	1. o cliente informa a conta e agência 2. o cliente informa a senha 3. executa caso de uso “Validação Conta” 4. o sistema exibe a lista de serviços do caixa eletrônico 5. o cliente seleciona a opção saque 6. o cliente digita o valor a ser sacado 7. o cliente informa a senha 8. verifica se o saldo é suficiente 9. executa caso de uso “Validação Conta” 10. o valor do saldo da conta é atualizado 11. as cédulas são liberadas no ***dispenser*** de cédulas

Fluxo Alternativo 1	1. O cliente cancela a operação por teclado finalizando o caso de uso
Fluxo Alternativo 2	1. O sistema informa a mensagem "Conta e/ou Senha inconsistente" 2. O sistema finaliza a tarefa por senha ou conta inexistente
Fluxo Alternativo 3	1. O sistema informa a mensagem "Saldo Insuficiente para Saque" 2. O sistema finaliza a tarefa por saldo insuficiente
Pós-condição	Cédulas disponibilizadas
R. Negócio	RN02

Quadro 5.4 - Exemplo de documentação do caso de uso realizar saque
Fonte: Elaboração da autora (2008).

Quais são as regras de negócio?

Quando você especifica uma regra de negócio você está especificando alguma condição, uma restrição, uma política ou uma norma que pode de alguma maneira interferir no processo que será realizado por seu projeto.

Regras de negócio mudam de uma empresa para outra, outras são comuns a várias empresas. Veja alguns exemplos de regras de negócio:

- no sistema de caixa eletrônico o cliente só poderá realizar o saque se e somente se: o saldo de sua conta corrente for superior ou igual ao valor a ser sacado. Se o saldo for inferior ao valor a ser sacado e se o cliente possui um valor de limite de crédito em sua conta corrente, então o saque não pode ser superior ao valor do saldo do valor limite da conta corrente.

- no sistema de estoque, na baixa do estoque quando a quantidade em estoque de um produto for inferior a quantidade de estoque mínimo deve ser gravada uma solicitação de pedido para esse item. A quantidade solicitada será a diferença de quantidade sobre a quantidade mínima.

As regras de negócio podem ser documentadas por meio de uma identificação e descrição. Cada regra pode estar ligada a um ou mais casos de uso, nesse caso o identificador deve ser anexado ao caso de uso.

Identificador	Descrição da regra de negócio
RN01	O número máximo de impressão de saldos no mês é de 3 saldos mensais
RN02	O cliente só poderá realizar o saque se e somente se: o saldo de sua conta corrente for superior ou igual ao valor a ser sacado. Se o saldo for inferior ao valor a ser sacado e se o cliente possui um valor de limite de crédito em sua conta corrente, então o saque não pode ser superior ao valor do saldo do valor limite da conta corrente.

Quadro 5.5 – Exemplo de documentação das regras de negócio
Fonte: Elaboração da autora (2008).

Sugestão para documentação dos casos de uso

É importante estruturar a sequência em que você vai organizar esta documentação. Existem metodologias que apoiam estas decisões como o **RUP** e o **Iconix**. Uma sugestão de documentação pode obedecer esta sequência:

- Se você não preencheu o documento de análise do problema e especificação de requisitos liste os requisitos funcionais e não funcionais.
- Documente o ator (lembra da tabela 5.1?)
- Insira o diagrama de casos de uso gerais do projeto.
- Uma tabela de documentação do caso de uso (tabelas 5.3 e 5.4) para cada caso de uso.
- Documente as regras de negócio (tabela 5.5).

Rational Unified Process (RUP) é definido como um processo de engenharia de software que oferece uma abordagem baseada em disciplinas possibilitando a atribuição de tarefas e responsabilidades no desenvolvimento de software (SOUZA; BRAGA, 2004).

Iconix - O ICONIX é um processo simplificado que unifica conjuntos de métodos de orientação a objetos em uma abordagem completa, com o objetivo de dar cobertura ao ciclo de vida do desenvolvimento de software (ROSENBERG; SCOTT, 1999).

E o que são pacotes?

O uso de pacotes permite a visualização mais organizada principalmente em sistemas grandes.

Se houver muitos casos de uso ou atores, você pode usar os pacotes de casos de uso para estruturar ainda mais o modelo de casos de uso. Um pacote de casos de uso contém vários atores, casos de uso, seus relacionamentos e outros pacotes.

Imagine que você tenha a seguinte lista de casos de uso:

- Cadastro de Clientes;
- Pedidos em Aberto;
- Gerenciamento de Vendas;
- Relatório de Data de Validade Vencidas;
- Cadastro de Produtos;
- Lista de Clientes;
- Lista de Produtos;
- Relatório de Produtos Mais Vendidos;
- Pedidos de Cancelamento.

O projetista pode agrupar casos de uso que apresentam aspectos comuns, como o tipo de função, atores que utilizam ou questões organizacionais. Observe a figura a seguir: os casos de uso foram divididos em três pacotes, e esse agrupamento facilita a visualização, principalmente em sistemas com um grande número de casos de uso.

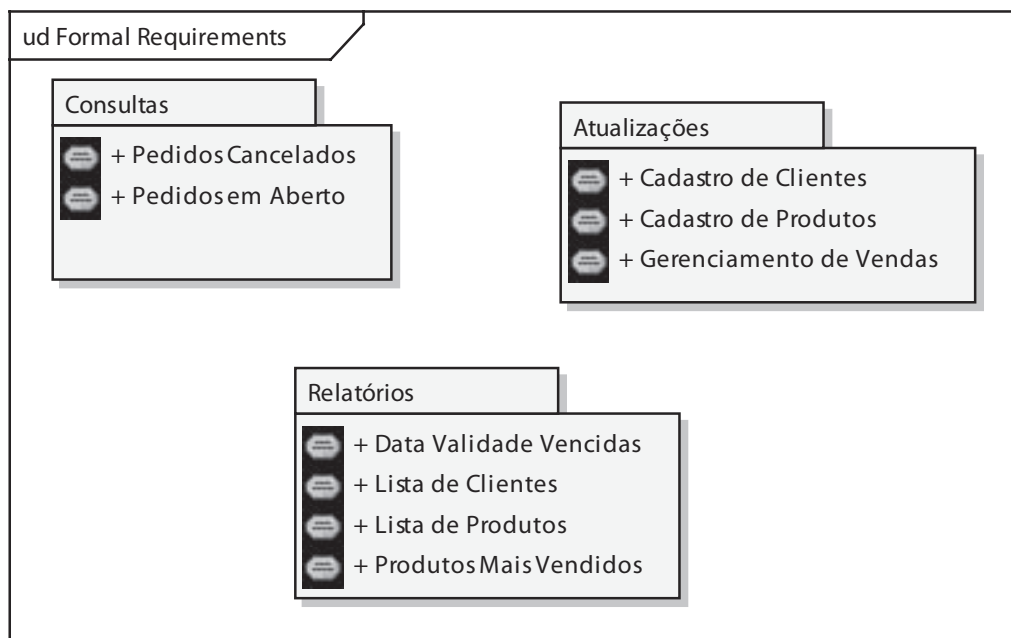



Figura 5.14 – Diagrama de pacotes
Fonte: Elaboração da autora (2011).

Quer conhecer mais ?

Para melhorar seus conhecimentos sobre esta unidade você pode ler:

- O capítulo 4 do livro: **Princípios de Análise e Projeto de Sistemas com UML**, de Eduardo Bezerra, publicado em 2002.

Na Midiateca você vai achar dois artigos interessantes abordando o assunto, sob os links:

- UML - Linguagem de Modelagem Unificada – Diagrama de caso de uso

Imagine que vocês foi contratado para modelar um sistema de imobiliária. Após a análise os requisitos funcionais foram assim definidos:

RF01 -> Requisito Funcional + numeração sequencial

Cadastro de clientes	RF001
Permitir a inclusão, alteração e exclusão de clientes para compra, venda ou aluguel de imóveis.	

Cadastro de corretores	RF002
Permitir o cadastro de corretores que vendem imóveis para a imobiliária.	

Cadastro de fiador	RF003
Cadastro dos fiadores usados pelos clientes permitindo incluir, alterar, excluir seus dados.	

Cadastro de imóveis	RF004
Cadastra todos os dados dos imóveis que são negociados na imobiliária. Imóveis podem ser para locação ou venda: casa, apartamento, quitinete, comercial.	

Alugar imóvel	RF005
Cadastra os dados de aluguel de um imóvel para um cliente, como data de locação, data de término de contrato, valor do aluguel.	
Vender imóvel	RF006
Cadastro das vendas realizadas pelos corretores permitindo incluir, alterar e excluir registros de venda.	

Gerar contrato	RF007
Gera o contrato para ser impresso no ato da venda ou aluguel do imóvel. A geração do contrato e dados deve ser automática.	

Emitir boleto bancário	RF008
Emite o boleto de cobrança para clientes que alugam imóveis com dados como valor do aluguel, IPTU, descontos e multas.	

Controle das comissões	RF009
Emite um extrato com os imóveis alugados ou vendidos indicando o valor da comissão do corretor.	

Busca de imóvel	RF010
Propiciar a realização da busca de imóvel por parâmetros informados como bairro, número de quartos, valor aproximado de aluguel.	

Gerar relatório de cobrança	RF011
Permitir a geração de um relatório com os imóveis alugados que estão com mensalidade atrasada.	

Cadastra pagamento	RF012
Deve ser permitido o registro do pagamento de aluguel de um imóvel.	

Gerar relatório	RF013
Gerar um relatório com quantidade de vendas e aluguéis realizados e desfeitos. Classificado por mês e ano.	

Cadastrar usuário do sistema	RF014
Permitir o cadastro de usuários do sistema, para que se possam definir níveis de acesso por meio de contas e senhas.	

Login de usuário	RF015
Efetuar login para identificar quem está usando o sistema e definir os acessos que ele possui.	
Cadastro de manutenções	RF016
Cadastra dados de manutenções feitas no imóvel armazenando o tipo de manutenção, o valor gasto, data da manutenção e uma breve descrição.	

Requisitos não funcionais

RNF01 -> Requisito Não Funcional + numeração sequencial

Tempo de resposta	RNF01
O tempo de resposta para consultas ao sistema, como a busca de imóvel, não devem ser inferiores a 5 segundos.	

Manutenibilidade	RNF02
O sistema deve ser construído obedecendo à visão de camadas facilitando futuras manutenções.	

Durante a análise perceberam-se as regras de negócio relacionadas ao funcionamento do processo na imobiliária:

Identificador	Descrição
RN01	O cliente deve ter fiador para contratar aluguel
RN02	CPF do cliente e fiador devem ser válidos.
RN03	O imóvel deve ser aprovado pelo gerente antes de ser cadastrado.
RN04	O crédito do cliente deve ser aprovado pelo gerente antes de efetivar o contrato de aluguel.
RN05	O valor da taxa do boleto, cobrada pelo banco, é adicionado no valor total do boleto do locador.
RN06	A multa por atraso no pagamento do aluguel é de 0,1% do valor do aluguel por dia de atraso.
RN07	O pagamento do boleto após o vencimento só pode ser efetivado em estabelecimento bancário conveniado à imobiliária.
RN08	As manutenções ou reformas feitas pelo locatário no imóvel são por conta própria, ou seja, não serão abatidas no valor da mensalidade.

Quadro 5.6 - Regras de negócio do sistema imobiliário
Fonte: Elaboração da autora (2008).

No sistema imobiliário foram identificados quatro atores:

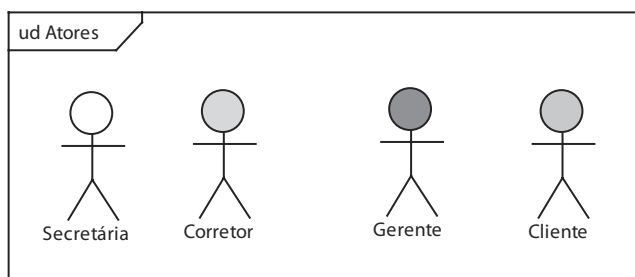



Figura 5.15 – Atores do sistema imobiliário
Fonte: Elaboração da autora (2008).

A partir dos requisitos funcionais foram definidos os casos de uso. Os casos de uso foram subdivididos em três pacotes Negociar Imóvel, Gerenciamento e Administração. Os casos de uso relacionados a cada pacote encontram-se inseridos no mesmo.

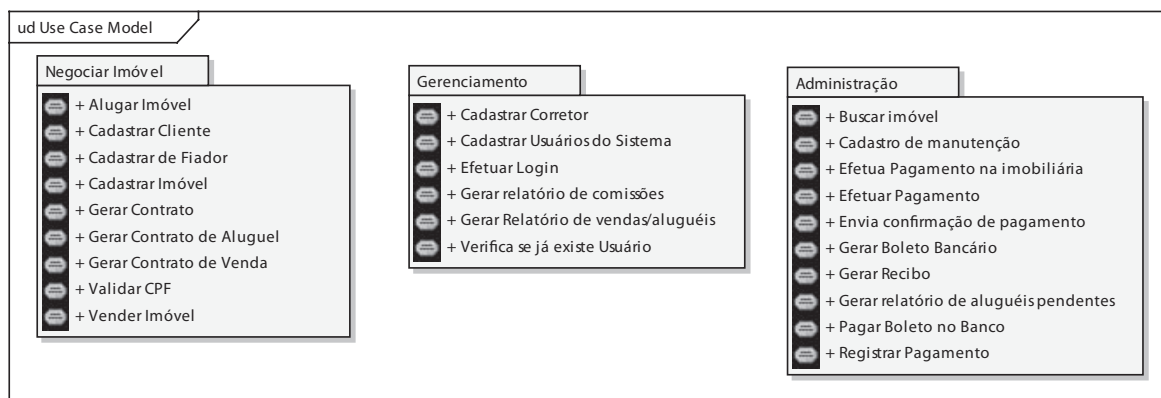



Figura 5.16 – Pacotes de casos de uso do sistema imobiliário
Fonte: Elaboração da autora (2008).

O diagrama de casos de uso pode ser feito de forma geral ou por pacotes, observe a seguir diagrama de casos de uso dos pacotes Gerenciamento e Negociar Imóvel:

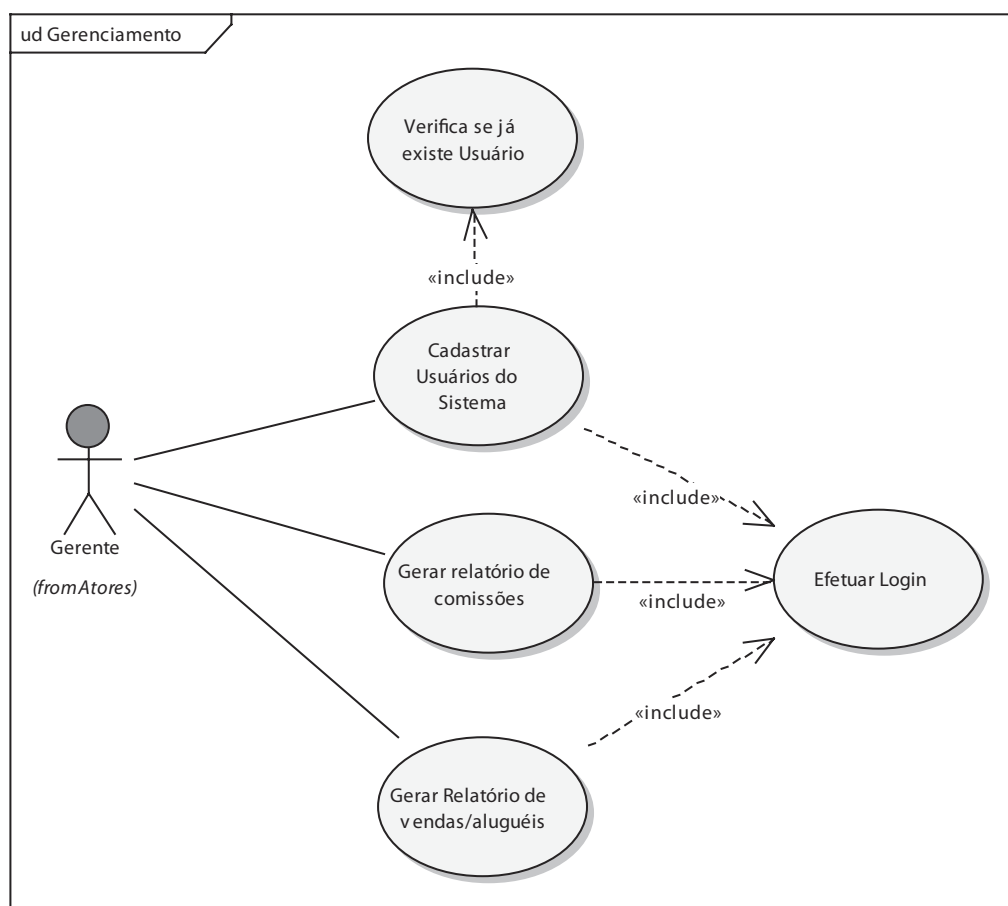



Figura 5.17 – Diagrama de casos de uso do pacote gerenciamento
Fonte: Elaboração da autora (2008).

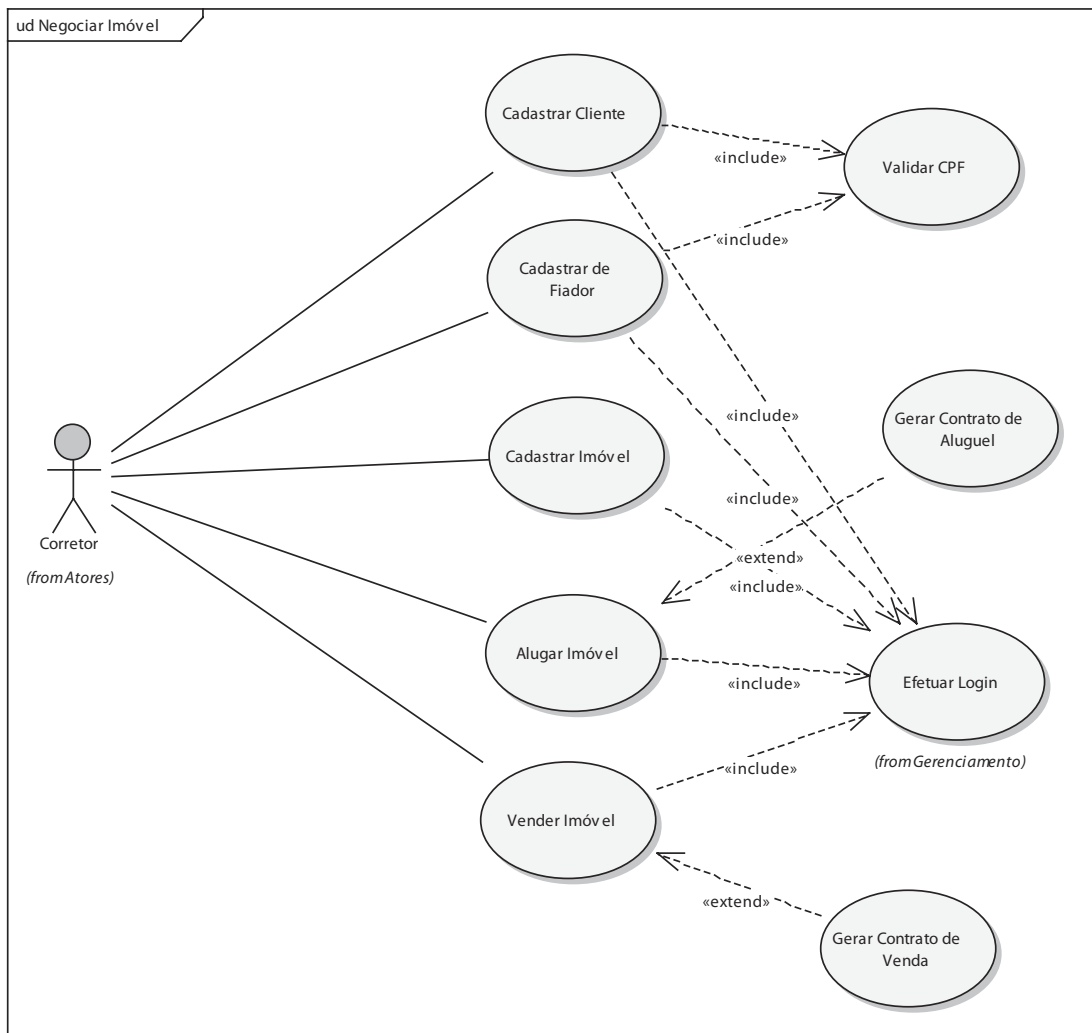



Figura 5.18 – Diagrama de casos de uso do pacote negociar imóvel
Fonte: Elaboração da autora (2008).

Definidos os diagramas você vai documentar cada caso de uso, como os exemplos apresentados a seguir:

Nome	Cadastrar Fiador
Identificador	CSU02
Descrição	O corretor cadastra as informações inerentes ao fiador de um cliente.
Ator Primário	Corretor
Precondição	Corretor logado no sistema
Fluxo Principal	1. Corretor digita dados do fiador no sistema. 3. Corretor Confirma cadastro. 4. Executa "Validar CPF". 5. Dados do Fiador são armazenados.
Fluxo Alternativo 1	1. Corretor seleciona botão Cancelar. 2. Os dados da janela são limpos.
Fluxo Alternativo 2	1. Corretor deseja alterar dados de algum fiador 2. Corretor seleciona um fiador cadastrado. 3. Sistema mostra informações do fiador. 4. Corretor altera informação 5. Executa "Validar CPF". 6. Corretor confirma a alteração 7. Dados do Fiador são armazenados
Fluxo Alternativo 3	1. Corretor deseja excluir fiador. 2. Corretor seleciona um fiador cadastrado. 3. Seleciona botão Exclusão. 4. Dados do fiador são excluídos do banco de dados.
Pós-condição	Dados do fiador atualizados
Regras de Negócio	RN02

Figura 5.19 - Documentação do caso de uso cadastrar fiador
Fonte: Elaboração da autora (2008).

Nome	Cadastrar Imóvel
Identificador	CSU04
Descrição	Efetua o cadastro do imóvel que ficará disponível na imobiliária.
Ator Primário	Corretor
Precondição	Corretor logado no sistema e o Proprietário do imóvel precisa estar cadastrado no sistema como cliente.
Fluxo Principal	1. Corretor informa código do cliente 2. Corretor insere dados do imóvel no sistema. 3. Corretor confirma cadastro.
Fluxo Alternativo 1	1. Corretor seleciona botão Cancelar. 2. Os dados da janela são limpos.
Fluxo Alternativo 2	8. Corretor deseja alterar dados do imóvel 9. Corretor seleciona um imóvel cadastrado. 10. Sistema mostra informações do imóvel. 11. Corretor altera informação 12. Corretor confirma a alteração 13. Dados do Imóvel são armazenados
Fluxo Alternativo 3	5. Corretor deseja excluir imóvel. 6. Corretor seleciona um imóvel cadastrado. 1. Seleciona botão Exclusão. 2. Dados do imóvel são excluídos do banco de dados.
Pós-condição	Imóvel atualizado no banco de dados.
Regras de Negócio	RN03

Figura 5.20 - Documentação do caso de uso cadastrar imóvel
Fonte: Elaboração da autora (2008).

Agora, para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas.

Síntese

Nesta unidade, você aprendeu a identificar os casos de uso mais significantes e seus principais elementos.

Durante o estudo também foram identificados os possíveis relacionamentos existentes entre atores e casos de uso. Você percebeu que o caso de uso estabelece a estrutura principal da modelagem permitindo uma comunicação fácil e precisa com o usuário, pois esta visão não inclui aspectos relacionados à implementação facilitando a interpretação por parte de desenvolvedor e usuários, identificando as ações necessárias ao sistema para que o usuário alcance a solução de suas necessidades.

A partir dos conceitos elementares você seguiu em direção à concepção do diagrama e da documentação necessária para o bom entendimento do modelo.

Atividades de autoavaliação

Leia com atenção os enunciados e, após, realize as questões propostas:

1) Assinale a afirmativa correta:

a) ( ) Um caso de uso procura apoiar a especificação de detalhes necessários à implementação do sistema.

b) ( ) O caso de uso documenta as ações necessárias, comportamentos e sequências visando atender as necessidades do usuário.

c) ( ) Em um caso de uso são necessários três elementos básicos: o caso de uso, a classe de controle e o ator.

d) ( ) O ator de um caso de uso representa apenas os usuários do sistema.

2) Classifique as questões em Verdadeira (V) ou Falsa (F).

a) ( ) Um ator pode ser representado por um organização ou mesmo um equipamento de *hardware*.

b) ( ) O uso de um nome pessoal para um determinado ator cria uma identificação pessoal do usuário com a especificação do caso de uso, facilitando sua validação com o usuário.

c) ( ) A relação existente entre um caso de uso e um ator pode ser uma relação de comunicação ou uma relação de extensão.

d) ( ) A relação existente entre dois atores pode ser somente uma relação de herança.

e) ( ) A relação existente entre dois casos de uso pode ser de extensão, inclusão e comunicação.

f) ( ) A relação existente entre dois casos de uso pode ser de extensão, inclusão e herança.

3) Relacione a primeira com a segunda coluna.

A. Regras de negócio

B. Ator

C. Cenário

D. Caso de uso

( ) DVDs em atraso pagam o valor da locação dos dias atrasados

( ) Atendente

( ) "A reserva de um DVD pode ser realizada pessoalmente, em que o cliente informa seu nome, o DVD desejado e a data da reserva. Caso o DVD não esteja reservado, o atendente realiza a reserva. O DVD pode ser reservado na página da videolocadora. Informando seu nome, o cliente seleciona o DVD e informa a data da reserva. A reserva é efetiva somente pelo envio da confirmação para o *e-mail* do cliente."

( ) Devolução de DVDs

( ) Fornecedor de DVDs

( ) Gerenciar compra de DVDs

( ) O cliente pode retirar no máximo 3 DVDs com devolução para 24 horas.

4) No texto a seguir, identifique:

a) Os requisitos funcionais

b) Os atores

c) Os casos de uso

d) Documente dois casos de uso

Requisitos funcionais:

- O projeto que você vai realizar será para a clínica pediátrica Bem-Estar e tem como objetivo principal a marcação de consultas médicas.
- O paciente pode realizar o agendamento da consulta pessoalmente ou por telefone. Em qualquer dos dois métodos os procedimentos são idênticos.
- O paciente solicita a consulta informando o nome do médico ou a especialidade desejada, posteriormente informa a data desejada. A atendente verifica a possibilidade de marcação da consulta (observando se o médico em questão possui horário vago para a data desejada). Se existe horário disponível, a atendente solicita ao paciente o tipo de convênio ou se é particular. Se for convênio, é verificado se é um convênio válido; se for particular, é informado o valor da consulta. A atendente atualiza a agenda com o nome do paciente e o tipo de consulta (convênio/particular). O tempo para cada consulta é de 20 minutos ou 10 minutos para retorno.
- A consulta pode ser uma consulta de retorno. Nesse caso, a atendente verifica se a data está ainda dentro do prazo de retorno de 15 dias. Neste caso a consulta é marcada na agenda.
- Caso o médico solicitado esteja indisponível, a atendente procura informar o nome de outro médico disponível naquele horário ou o próximo horário disponível.
- O paciente pode telefonar desmarcando a consulta, nesse caso o nome do paciente é riscado da agenda, ficando o horário vago novamente.
- Ao chegarem na clínica, os médicos recebem as fichas dos pacientes separadas previamente. Se for paciente novo, a ficha contém somente o nome do paciente e o telefone. As fichas são classificadas por ordem de horário.
- Se o paciente já possui cadastro, o mesmo é convidado a adentrar no consultório do médico. A partir desse momento, o médico solicita informações procedimentais para o futuro diagnóstico, preenchendo a ficha do paciente. Finalizada a consulta, o paciente é liberado e a ficha é recolhida pela atendente, sendo novamente arquivada.
- Se o paciente for novo, a atendente solicita o preenchimento da ficha cadastral com dados pessoais.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar:

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** guia do usuário. Rio de Janeiro: Campus, 2000. (Ler capítulos 16 e 17.)

UNIDADE 6

Modelagem de classes

Objetivos de aprendizagem

- Identificar o papel do diagrama de classes no processo de análise.
- Conhecer e reconhecer termos técnicos, conceitos e relacionamentos utilizados durante a construção do diagrama de classes.
- Identificar as possíveis classes de um projeto.
- Utilizar o diagrama de classe com a notação UML na construção de um modelo de projeto.

Seções de estudo

Seção 1 O que são objetos e classes de objetos?

Seção 2 Quais são as responsabilidades das classes?

Seção 3 Como ocorrem os relacionamentos entre objetos?

Seção 4 Como ocorre a divisão das classes do modelo de análise?

Seção 5 O que é diagrama de objetos?

Para início de estudo

Você já estudou um dos aspectos mais fortes do projeto, que são os casos de uso. Eles permitem aos atores a visualização de resultados esperados, relatórios e processamentos.

De qualquer maneira, a possibilidade da existência dessa colaboração depende de aspectos estáticos e dinâmicos do sistema. Entre objetos do sistema, o aspecto dinâmico está fortemente ligado à troca de mensagens, enquanto o aspecto estático mostra como o sistema "está estruturado internamente" e passa por três níveis de abstração:

- o primeiro deles, o modelo de classe de domínio, representa a classe de domínio sem se preocupar com detalhes sobre a tecnologia;
- o segundo nível é o modelo da classe de especificação que acrescenta detalhes relacionados à solução do *software* escolhida pelo modelo do domínio;
- o terceiro nível, o modelo de classe de implementação é uma extensão do modelo de especificação. Ocorre neste nível a implementação em alguma linguagem de programação.

Para você entender a utilização desses níveis de abstração, é necessário conhecer conceitos e relacionamentos vinculados ao modelo de classes. O modelo de classes é um dos modelos mais ricos em termos de notação e concentra o cerne estático de todo o projeto.

Então, que tal escalar o mundo conceitual das classes?

Seção 1 – O que são objetos e classes de objetos?

De acordo com Pádua (2001), as entidades de domínio são representadas na modelagem orientada a objetos por objetos. O objeto representa uma entidade que pode ser física ou de *software*.

Na realidade, um objeto sempre é descrito por meio do estado, do comportamento e da identidade.

- A **identidade** é uma propriedade que irá distingui-lo dos demais.
- O **estado** de um objeto compreende características herdadas ou distintas que contribuem para que se torne único.
- O **comportamento** de um objeto define como se darão sua ação e reação a estímulos em termos de mudanças de estados e mensagens.

Classe Cliente

Identidade
Cliente

Estado

Nome
Código
Endereço
Telefone
Permissão

Comportamento:
Adicionar Cliente( )
Excluir Cliente( )
Consultar Cliente( )

Você pode dizer também que **um objeto é uma pessoa, um lugar ou um sistema**, como mostra a figura a seguir.

Os objetos também são chamados de instância.

Figura 6.1 – Exemplos de objetos
Fonte: Elaboração da autora (2010).

Segundo Furlan (1998), **classe é a representação de um conjunto de coisas reais ou abstratas que são reconhecidas como sendo do mesmo tipo por compartilhar as mesmas características de atributos, operações, relações e semântica.**

Um objeto é sempre uma instância de uma classe. Quando você fala de uma classe, está falando também de seus objetos.

Booch (2000) define classe como um conjunto de objetos que compartilham estrutura e comportamento comuns.

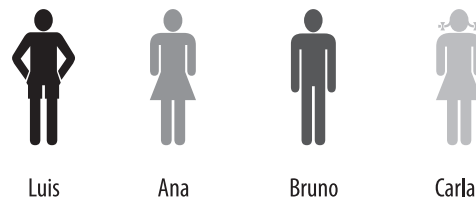



Figura 6.2 – Classe Clientes
Fonte: Elaboração da autora (2010).

Conforme você pode observar na figura 6.2, Luis, Ana, Bruno e Carla são objetos ou instâncias da classe. Mas qual a notação utilizada para identificar uma classe?

A classe é representada por um retângulo dividido em três compartimentos que contêm o nome da classe, os atributos e as operações.

Figura 6.3 – Notação da classe
Fonte: Elaboração da autora (2008).

O que são atributos?

O atributo é a descrição dos dados armazenados pelos objetos de uma classe. O atributo de uma classe está associado a um conjunto de valores que o atributo pode assumir.

Está correto afirmar que **os atributos não têm comportamento**. Cada valor de um atributo é particular para um dado objeto.

Uma classe pode ter qualquer número de atributos ou nenhum atributo. Os atributos são sempre individuais e cada objeto da classe possui seus próprios atributos.

Recordando do projeto da videolocadora, você pode definir as classes?

É possível detectar imediatamente três classes nesse projeto:

- Classe Cliente (dados e métodos do cliente);
- Classe Filmes (dados e métodos dos filmes);
- Classe Locação (dados e métodos sobre a locação).

O que são operações?

As operações implementam serviços que podem ser solicitados por algum objeto da classe para modificar o comportamento, ou seja, todos os objetos da classe vão compartilhar essas operações.

As operações são executadas quando um objeto recebe uma mensagem de outro objeto. Entretanto, existem situações em que uma classe pode não ter nenhuma operação ou mesmo ter várias operações.

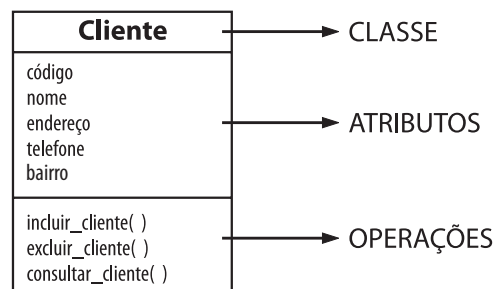



Figura 6.4 – Classe Cliente
Fonte: Elaboração da autora (2008).

Observe que:

- **para nomear a classe,** você deve escolher um substantivo, por exemplo: fornecedor, produtos, cliente;
- **para nomear uma operação,** faça uso de um verbo ou de um verbo mais um substantivo. A escolha do nome da operação deve possuir um nome que indique o resultado da operação (Cancelar_Fornecedor) acrescentando-se os parênteses "( )" ao final do nome da operação;
- **para nomear atributos,** utilize substantivos simples ou verbos substantivados. Se você for construir uma classe para um sistema de controle de estoque chamada Produtos, os atributos da classe Produto podem ser código, descrição_Produto ou unidade;
- para os três casos, ao atribuir nomes, utilize uma definição concisa e clara, não dando margem a interpretações errôneas.

Quais seriam os atributos para a classe Filmes?

Você poderia ter nesse caso:

- Nome_;
- Código_;
- Diretor_;
- Duração_;
- Ator_Principal1;
- Ator_Principal2;
- Tipo_;
- Idiomas_.

Agora tente imaginar as possíveis operações:

- Incluir_Filme;
- Excluir_Filme;
- Consultar_Filme;
- Listar_Filme.

Observe algumas classes que fariam parte do domínio do sistema da videolocadora.

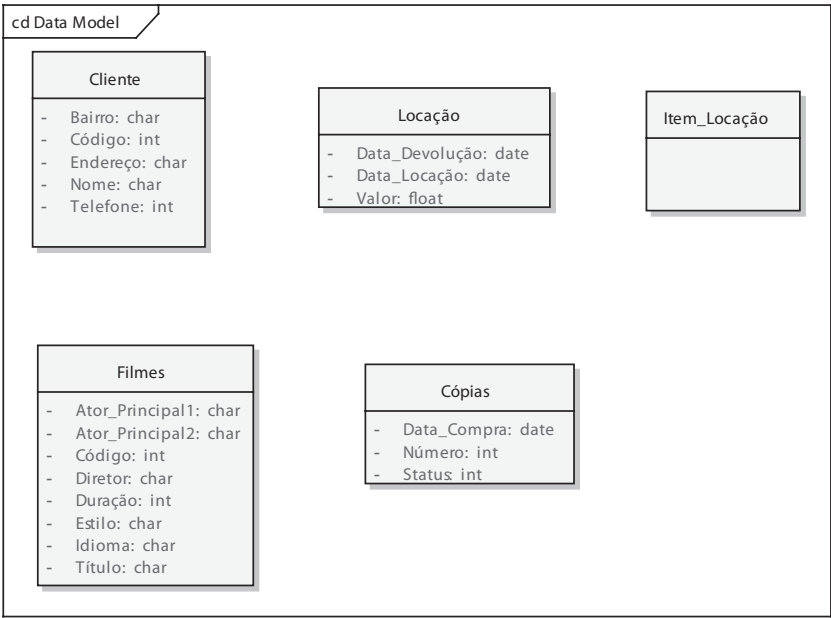



Figura 6.5 – Possíveis classes do sistema de videolocadora
Fonte: Elaboração da autora (2008).

A notação de classes apresenta três compartimentos, porém as classes podem ser apresentadas em diferentes níveis de abstração, como apresentado na seguinte figura.

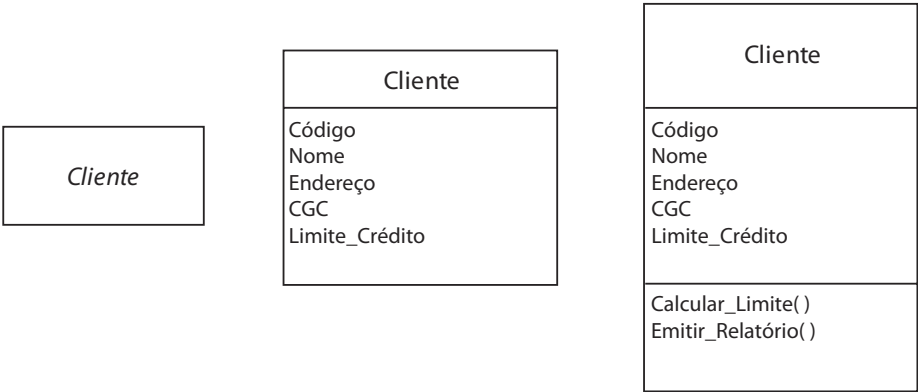



Figura 6.6 – Abstrações das classes
Fonte: Elaboração da autora (2008).

Seção 2 – Quais são as responsabilidades das classes?

Uma responsabilidade é um contrato ou obrigações de uma determinada classe, ou seja, representam o conhecimento e as ações que possibilitam que as classes cumpram seu papel nos casos de uso (PÁDUA, 2001).

Por exemplo: uma classe Cliente é responsável pelo conhecimento sobre os dados pessoais (código, nome, endereço etc.) do cliente, além de ser responsável por incluir, excluir e consultar os dados do cliente.

Mas definir as classes existentes em um futuro sistema e suas responsabilidades não é uma tarefa fácil. O mais adequado é você procurar o apoio de métodos reconhecidos que procuram facilitar sua realização. Para isso, existem dois métodos popularizados para essa etapa, que são: os cartões CRC e a análise dos casos de uso.

a) Os cartões Classes Responsabilidades e Colaborações (CRC) – O uso dos cartões CRC identifica as responsabilidades dos atributos e das operações apoiando a identificação de classes ou de candidatos às classes.

Cada ficha corresponde a uma classe, contém o nome da classe e duas colunas com descrição de suas responsabilidades e colaborações. As colaborações representam outras classes que interagem com a classe descrita para o cumprimento de suas responsabilidades.

O uso do cartão CRC é realizado com o envolvimento de toda a equipe. Para isso, uma sessão é organizada. Para realizar a sessão:

- o primeiro passo é a escolha do grupo de pessoas que representará um cenário, ou seja, é exatamente o cenário do domínio do problema;

- na segunda etapa, cada participante é associado a uma **classe**. Assim, cada pessoa passa a pertencer àquela classe e todo o cenário será encenado pelos participantes. Aos poucos cada cartão é preenchido com as responsabilidades e os colaboradores.

Para cada classe de objeto identificada dentro do cenário, é criado um cartão CRC.

Durante a sessão pela exploração dos cenários, é comum que novos cartões sejam criados pela descoberta de novas classes.

Observe o cartão CRC criado a partir do seguinte cenário:

- O balconista faz a abertura da venda.
- O balconista registra os itens de venda podendo inserir novos itens, excluí-los e editá-los.
- O sistema totaliza a venda para o cliente.
- O sistema calcula os impostos sobre a venda.
- O balconista encerra a venda.

Nome da Classe: Venda	
Responsabilidades	Colaborações
Inserir/excluir item de venda	Item de venda
Editar item de venda	Estoque
Totalizar venda	Mercadoria
Calcular impostos	Mercadoria

Quadro 6.1 - Exemplo de um cartão CRC
Fonte: Elaboração da autora (2008).

Você pode documentar as responsabilidades no próprio diagrama de classes. Neste caso, insira a descrição na forma de texto no final da caixa da classe, ou seja, na notação gráfica você tem mais um compartimento logo abaixo das operações.

b) Análise dos casos de uso – Se você optar por utilizar a análise de casos de uso para identificar as classes candidatas e suas responsabilidades, é importante analisar os casos de uso e todos os seus fluxos (principais e alternativos).

Mas como funciona esse método? Bom, a partir da análise são identificados os substantivos existentes nos casos de uso e os sinônimos são eliminados.

Muitas vezes, um substantivo pode ser um ator, o qual deve ser eliminado. Assim, os nomes que permaneceram são as classes candidatas.

Observe o exemplo a seguir:

Pádua (2001) mostra a análise de um estudo de caso para um sistema de vendas de uma pequena mercearia. Ao analisar o estudo de caso procurando as classes candidatas, os atores são selecionados em itálico, o sistema aparece em negrito e os substantivos em sublinhado.

O *balconista* faz a abertura da venda.

O *balconista* registra os itens vendidos informando código do produto e quantidade do item.

O **sistema** totaliza a venda para o cliente.

O *balconista* encerra a venda.

O **sistema** emite o ticket para o cliente.

O *balconista* registra a forma de pagamento.

O **sistema** faz a baixa no estoque das mercadorias vendidas.

Ao analisar o caso de uso, você deduz a possível lista de classes candidatas, operações e atributos:

Classe Candidata	Análise
Abertura	Operação
Venda	Provável classe
Item vendido	Provável classe – melhor item de venda
Código do produto	Atributo do item da venda
Quantidade	Atributo do item da venda
Cliente da mercearia	Entidade fora do escopo do produto
Ticket	Relatório
Forma de pagamento	Atributo da venda
Baixa	Operação
Estoque	Possível classe
Mercadoria	Possível classe

Quadro 6.2 – Exemplo lista de classes
Fonte: Elaboração da autora (2008).

Observe que o item **abertura** pode ser descrito como uma operação, assim como **venda** passa a ser uma possível classe. Já **código do produto** tem as características de um atributo. Note que já é possível identificar inclusive a qual classe o atributo pertence, como no caso da **quantidade**.

A partir da identificação é possível determinar no exemplo as classes candidatas do sistema de controle de mercearia:

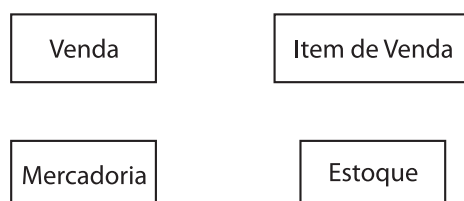



Figura 6.7 – Classes candidatas
Fonte: Elaboração da autora (2008).

Apesar da possibilidade de utilizar essas técnicas, o grande universo de informações do domínio do problema muitas vezes dificulta a identificação. Existem algumas dicas que podem ser utilizadas para descobrir as classes de um domínio de aplicação observando os seguintes elementos:

- informações que devem ser armazenadas, transformadas, analisadas ou manipuladas de alguma forma;
- outros sistemas e **terminadores externos** que se comunicam ou interagem com o sistema em questão, atentando para as informações que estão sendo trocadas;
- dispositivos físicos com os quais o sistema em estudo terá de interagir, sem considerar a tecnologia para implementar o sistema em si;
- modelos, bibliotecas de classe e componentes gerados em projetos anteriores.

Coad e Yourdon (1992) também sugerem alguns indicadores:

- coisas que são parte do domínio de informação do problema;
- ocorrências ou eventos que precisam ser registrados e lembrados pelo sistema;
- papéis desempenhados pelas diferentes pessoas que interagem direta ou indiretamente com o sistema;
- locais físicos ou geográficos e lugares que estabelecem o contexto do problema;
- unidades organizacionais (departamentos, divisões etc.) que possam ser relevantes para o sistema.

Lembra-se do projeto da Clínica Bem-Estar visto na unidade 2, atividade 3, de autoavaliação? Que tal definir as possíveis classes candidatas?

- Classe Paciente;
- Classe Médico;
- Classe Convênio;
- Classe Laboratório;
- Classe Agenda;
- Classe Ficha_Médica.

Agora quais os atributos que você considera pertinentes para a classe Convênio?

Você pode listar para a classe convênio atributos como:

- Código_Convênio;
- Nome_Convênio;
- Telefone_Convênio;
- Características_Convênio;
- Status_Convênio.

Se você listar os possíveis atributos da classe Médico:

- Nome_Médico;
- CRM_Médico;
- Endereço_Médico;
- Telefone_Médico;
- Celular_Médico;
- Especialidades_Médico;
- Horário_Médico.

Seção 3 – Como ocorrem os relacionamentos entre objetos?

Um relacionamento representa a interação entre as classes e os objetos, ele apoia o refinamento das classes.

Existem diferentes tipos de relacionamentos possíveis entre as classes identificadas. Os mais importantes são as associações, as dependências e as generalizações.

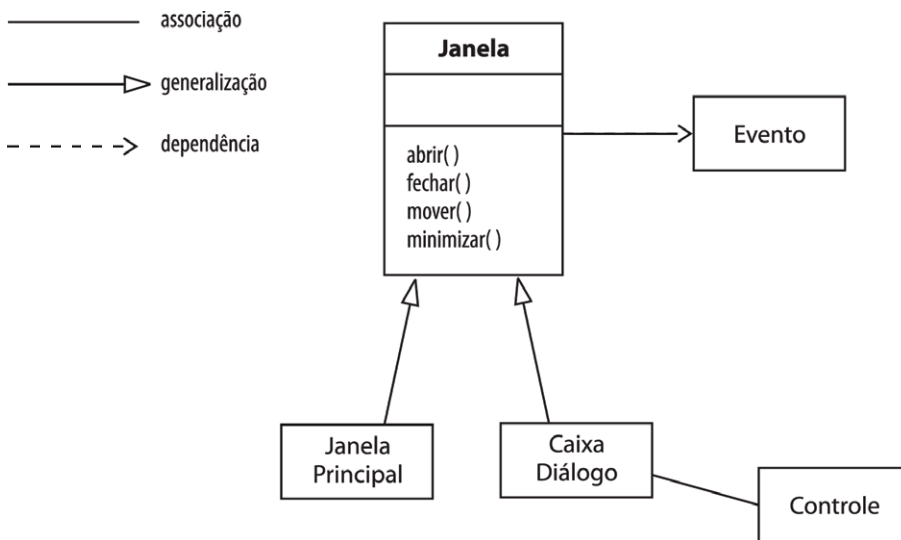



Figura 6.8 – Relacionamento entre objetos
Fonte: Adaptado de Booch (2000).

Conheça então, a partir de agora, as principais características sobre os tipos mais comuns de relacionamentos entre objetos.

a) Relacionamento de associação – Segundo Furlan (1998), associação é uma relação que descreve um conjunto de vínculos entre elementos de modelo: quando duas classes ou mesmo uma classe, consigo própria, apresenta interdependência; ou quando uma determinada instância de uma das classes origina ou se associa a uma ou mais instâncias da outra classe, você pode dizer que se tem um relacionamento de associação.

Figura 6.9 – Relacionamento de associação entre as classes
Fonte: Elaboração da autora (2008).

O que é multiplicidade dentro desse tipo de relacionamento?

Quando se menciona associações, é possível representar a quantidade de objetos aos quais o outro objeto está associado. Um exemplo prático: um projeto existe sem que seja alocado um funcionário para esse projeto? Quantos projetos podem ser alocados para cada funcionário? Quantos funcionários podem ser alocados para cada projeto?

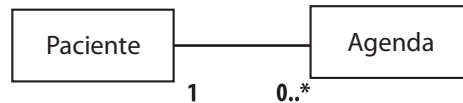
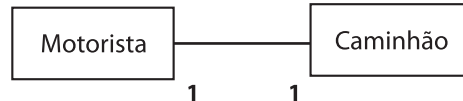
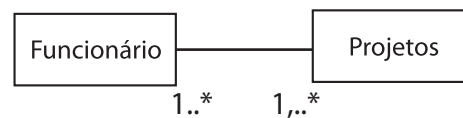
Na notação UML, isso é chamado de **multiplicidade**. Alguns autores também utilizam o termo **cardinalidade**. Mas como representá-lo em um diagrama?

Observe então a tabela a seguir que apresenta essa simbologia:

Nome	Simbologia
Apenas um	1
Zero ou muitos	0..*
Um ou muitos	1..*
Zero ou um	0..1
Intervalo específico	$1_{i..}1_{s}$

Quadro 6.3 – Simbologia UML
Fonte: Elaboração da autora (2008).

Veja alguns exemplos de multiplicidade:

- Um paciente possui nenhum ou vários agendamentos.

- Em uma empresa de transporte, um motorista dirige apenas um caminhão, e cada caminhão pode ser dirigido por apenas um motorista.

- No terceiro exemplo, um funcionário deve estar locado a um ou mais projetos. E cada projeto tem pelo menos um funcionário alocado.

Imagine um projeto para uma mercearia em que se pretende controlar os fornecedores, seus produtos e os pedidos de compra de produtos.

Imediatamente você identifica pelo menos três classes candidatas:

- Classe Fornecedor (que vai armazenar dados como endereço, nome, telefone etc.);

- Classe Produtos (que deve armazenar dados como código, descrição, unidade etc.);

- Classe Item de Compra (que deve armazenar preço, quantidade comprada etc.).

Você tem a classe Fornecedor, que pode ter de 0 a n produtos (ou seja, o fornecedor oferece à mercearia vários produtos para compra; veja o exemplo de uma revenda de bebidas que possui diferentes tipos de refrigerante e cerveja). Os produtos, por sua vez, podem ter de 0 a n fornecedores (posso ter mais de um fornecedor para o mesmo refrigerante).

Cada item de compra pode ter apenas um fornecedor, mas cada fornecedor pode ter de 0 a n itens de compra (no momento da compra vou ter um fornecedor apenas para aquele determinado item para aquela determinada nota fiscal).

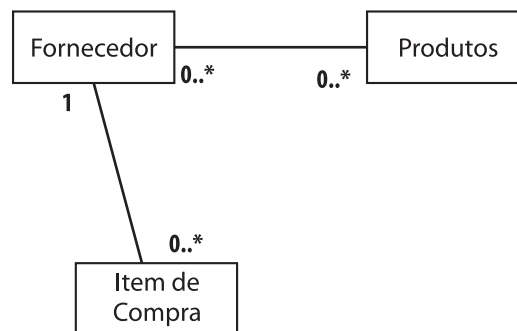



Figura 6.10 – Multiplicidade
Fonte: Adaptado de Pádua (2001).

Há várias maneiras de nomear associações. No entanto, você deve escolher o nome pensando na descrição da natureza do relacionamento.

Prefira o uso de um verbo ou uma frase verbal. Além de indicar um nome, você pode ainda indicar a direção da leitura da associação inserindo um triângulo de orientação.

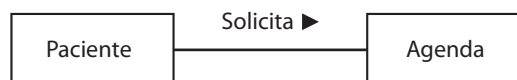



Figura 6.11 – Nomeando uma associação
Fonte: Elaboração da autora (2008).

Além desses recursos, observe que a classe, ao participar de uma associação, recebe um papel específico que é utilizado em um dos lados de uma associação com a finalidade de indicar qual o papel que a classe a seu lado apresenta para a classe do lado oposto.

Um papel define o propósito ou a capacidade de uma classe. Utilize substantivos para indicar os papéis (BEZERRA, 2002).

Na figura 6.12, há a classe Pessoa desempenhando o papel de funcionário na associação com a classe Banco, que desempenha o papel de empregador.

Figura 6.12 – Papéis em uma associação
Fonte: Elaboração da autora (2008).

O que significa agregação para o relacionamento de associação?

A agregação é um caso particular da associação. A agregação indica que uma das classes do relacionamento é uma parte ou está contida em outra classe (GUEDES, 2006). Mas as duas classes estão no mesmo nível, ou seja, não existe uma classe mais importante do que a outra na associação.

As palavras-chave usadas para identificar uma agregação são: **consiste em**, **contém** ou **é parte de**. Outra dica importante é que as partes não morrem obrigatoriamente com o todo, e uma mesma parte pode estar em mais de um **todo**. Graficamente você vai representar a associação de agregação por uma linha e um diamante aberto na extremidade.

Observe o exemplo de uma transportadora. Pode-se dizer que a empresa tem departamento, e a transportadora contém caminhões.

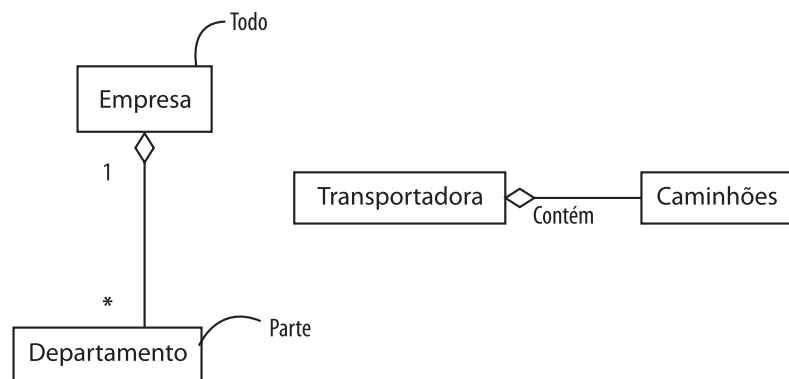



Figura 6.13 – Agregação
Fonte: Booch (2000).

A **composição** é um tipo especial de agregação em que a multiplicidade do lado "todo" é sempre 1. As partes vivem e morrem obrigatoriamente com o "todo". Uma mesma parte não pode estar em mais de um "todo". Os objetos da classe Parte não existem de forma independente da classe Todo.

A composição, segundo Rumbaugh (1994), é um tipo forte de associação em que um objeto agregado é composto de vários objetos componentes.

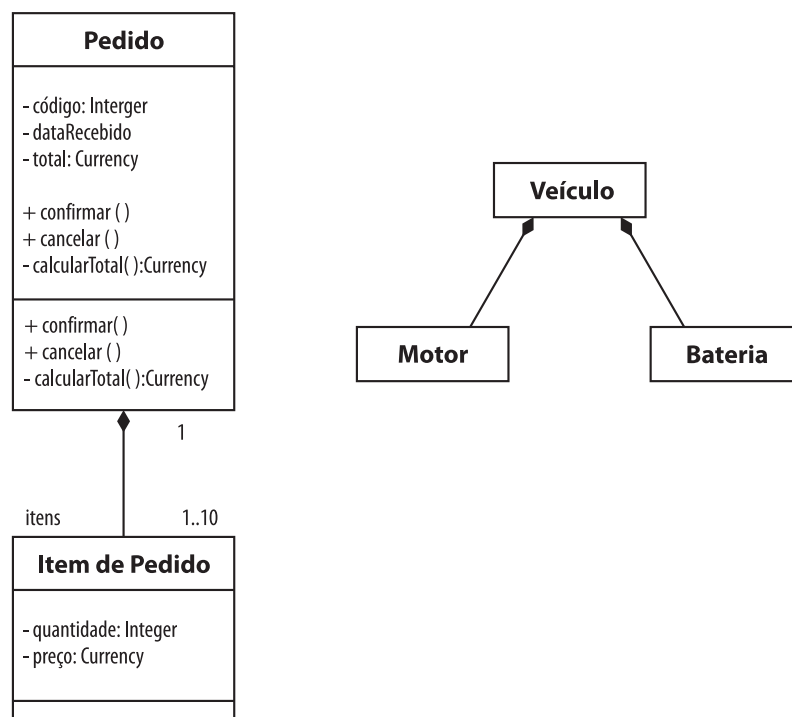



Figura 6.14 – Composição
Fonte: Elaboração da autora (2008).

Na representação das classes, de acordo com a figura 6.14, a associação de composição exprime que o Item de Pedido não existe sem o Pedido, ou seja , o Item de Pedido não existe de forma independente no sistema.

As classes Motor e Bateria nesse exemplo não existem de forma independente da classe Veículo. Elas fazem parte do todo "Veículo".

A figura 6.15 é um exemplo de classes para um sistema de controle de turmas em uma universidade. Nesse exemplo, a universidade oferece cursos aos alunos. Os cursos, por sua vez, oferecem disciplinas ministradas por professores aos alunos matriculados.

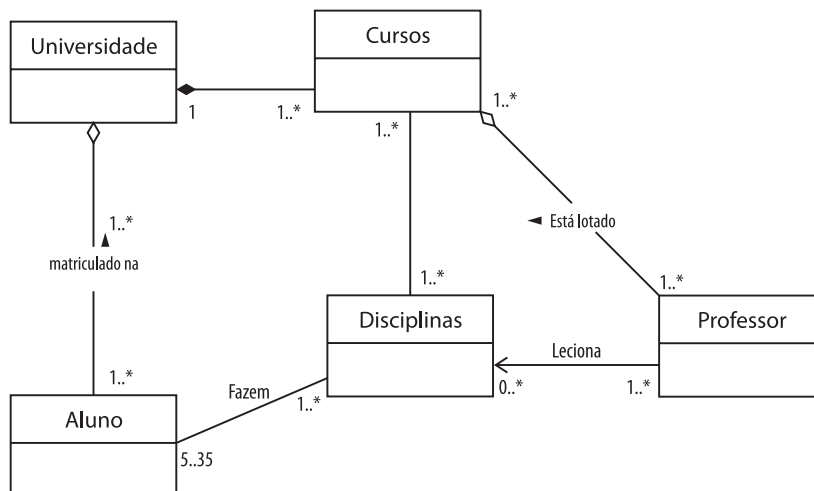



Figura 6.15 – Relacionamentos entre classes
Fonte: Elaboração da autora (2008).

As classes Universidade e Cursos possuem um relacionamento de composição, ou seja, a classe Curso não existe sem a classe Universidade. Se a classe Universidade for extinta, automaticamente a classe Cursos deixa de existir.

Já as classes Aluno e Professor apresentam um relacionamento de agregação, pois fazem parte de outra classe. A classe Aluno está contida na classe Universidade e a classe Cursos contém a classe Professor.

Para Bezerra (2002), as classes associativas estão ligadas a associações e não a outras classes. Simplificando, existem situações na análise em que atributos e operações são partes do relacionamento como um todo e não de cada uma das classes envolvidas. Nesse caso, em vez de se associarem esses atributos/ operações a um participante, é criada uma classe associativa que absorve esses atributos/operações.

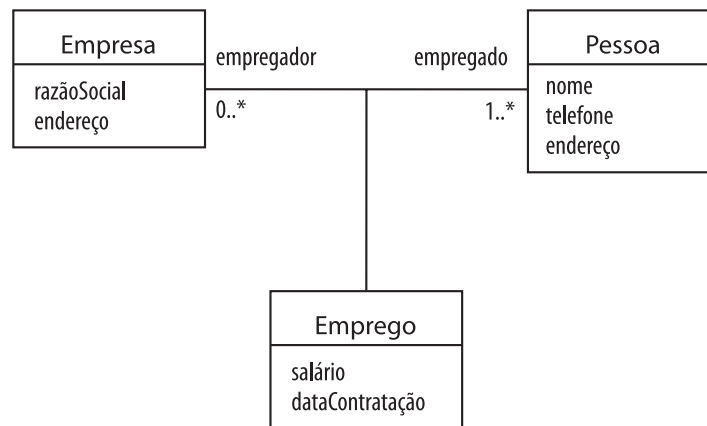



Figura 6.16 – Classe associativa
Fonte: Pádua (2001).

No exemplo ilustrado na figura 6.16, você vê uma situação em que uma pessoa trabalha em várias empresas e uma empresa tem vários empregados. Os atributos salário e dataContrataçao não pertencem à classe Empresa nem à classe Pessoa (que mantém dados cadastrais do empregado). Neste caso, uma classe associativa Emprego foi criada para comportar esses atributos para cada par (empregado/empregador).

b) Relacionamento de dependência – A dependência indica a ocorrência de um relacionamento entre dois ou mais elementos, no qual uma classe Cliente é dependente de algum serviço da classe Fornecedora.

Quando você precisar indicar que um item depende de outro, utilize o relacionamento de dependência.

Bezerra (2002) indica situações que levam a um relacionamento de dependência, como:

- **dependência por atributo** – a classe A possui um atributo cujo tipo é B;
- **dependência por variável global** – a classe A utiliza uma variável global cujo tipo é B;
- **dependência por variável local** – A possui alguma operação cuja implementação utiliza uma variável local do tipo B;
- **dependência por parâmetro** – A possui pelo menos uma operação que possui pelo menos um parâmetro cujo tipo é B.

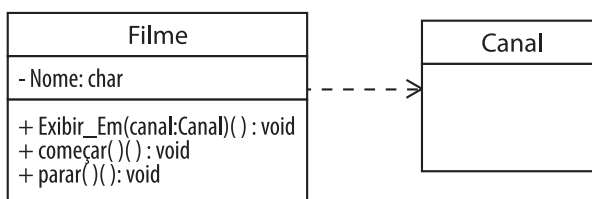



Figura 6.17 – Dependência
Fonte: Elaboração da autora (2008).

Para que as operações da classe Filme sejam executadas, a existência da classe Canal é fundamental, pois existe uma dependência de parâmetros entre as classes.

c) Relacionamento de generalização – A generalização é um relacionamento entre itens gerais (superclasses) e tipos mais específicos desses itens (as subclasses). A subclasse herda as propriedades da superclasse, principalmente atributos e operações.

Um relacionamento de especialização/generalização indica que objetos do elemento especializado (subclasse) podem substituir os objetos do elemento generalizado (superclasse). A subclasse tem todos os atributos e as operações da superclasse, porém pode ter outros atributos e operações.

Imagine a seguinte situação: você tem uma classe em seu sistema chamada Funcionários. Essa classe possui atributos como nome, endereço, entre outras informações. Existe, no entanto, um tipo de funcionário chamado motorista.

Esse funcionário possui atributos específicos como itinerário e horário das rotas. Os motoristas fazem parte da classe Funcionários, mas por suas características específicas formam outra classe. Neste caso, a classe Motorista herda as características da classe Funcionário.

Segundo Guedes (2006), a generalização, também conhecida como herança, pode ser simples ou múltipla.

- **Herança simples** – A subclasse herda estrutura e ou comportamento de uma única superclasse.
- **Herança múltipla** – A subclasse herda a estrutura e o comportamento de mais de uma superclasse.

Mas talvez você esteja se questionando: o que uma subclasse pode herdar de uma superclasse?

A subclasse pode herdar atributos, operações e relacionamentos. Além das características herdadas, a subclasse possui atributos, operações ou relacionamentos adicionais.

Na figura 6.18, a classe Imóvel possui atributos e operações comuns, como área, endereço e IPTU. Mas Apartamento possui características específicas, como valor do condomínio e fundo de reserva. O mesmo acontece com a classe Casa. Assim, as subclasses possuem todas as características da superclasse, e, além disso, possuem características específicas de cada subclasse. As subclasses são especializações da superclasse Imóvel.

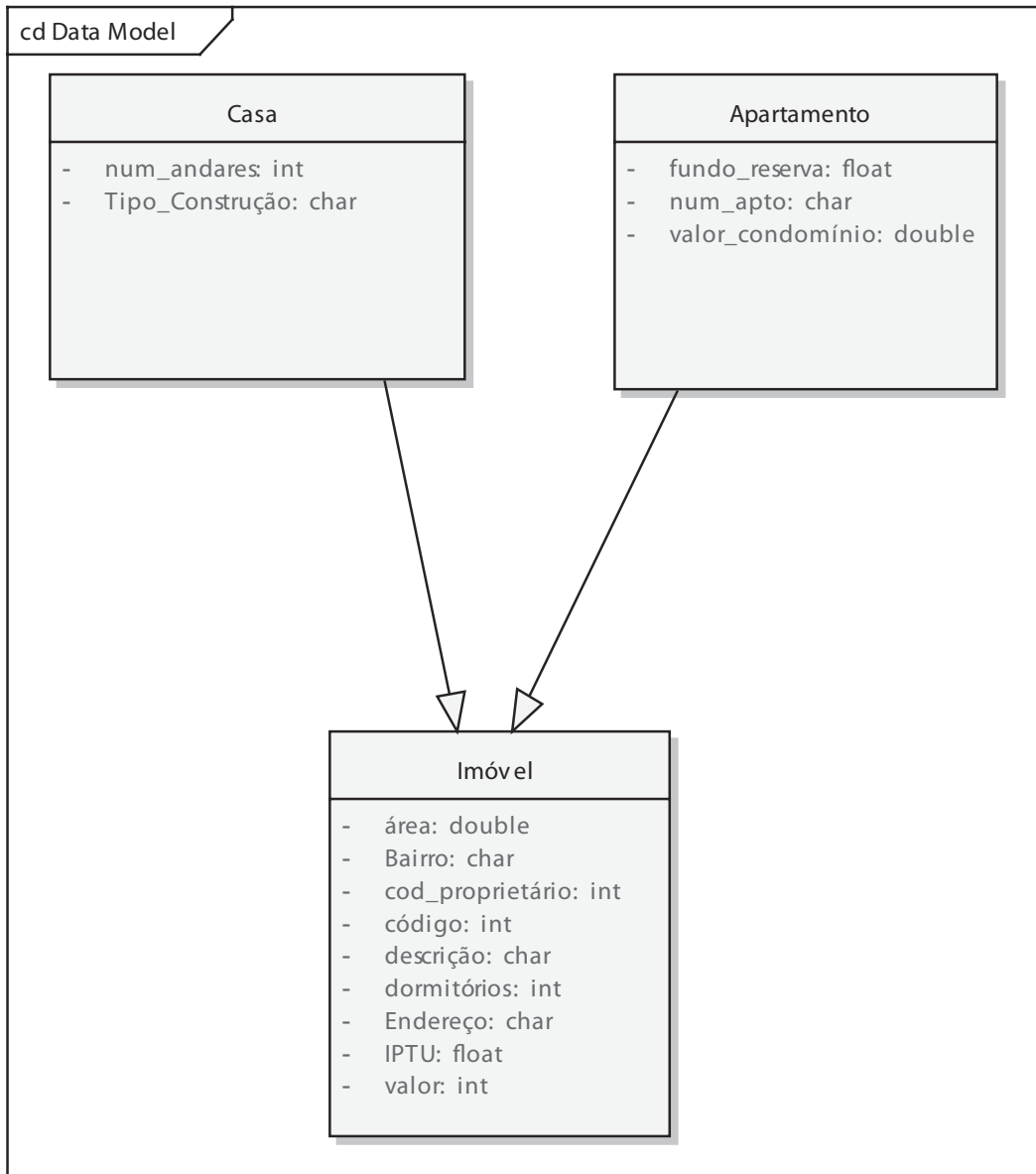



Figura 6.18 – Relacionamento de herança
Fonte: Elaboração da autora (2008).

Agora, relembrando o sistema da videolocadora, observe os relacionamentos entre as classes candidatas propostas:

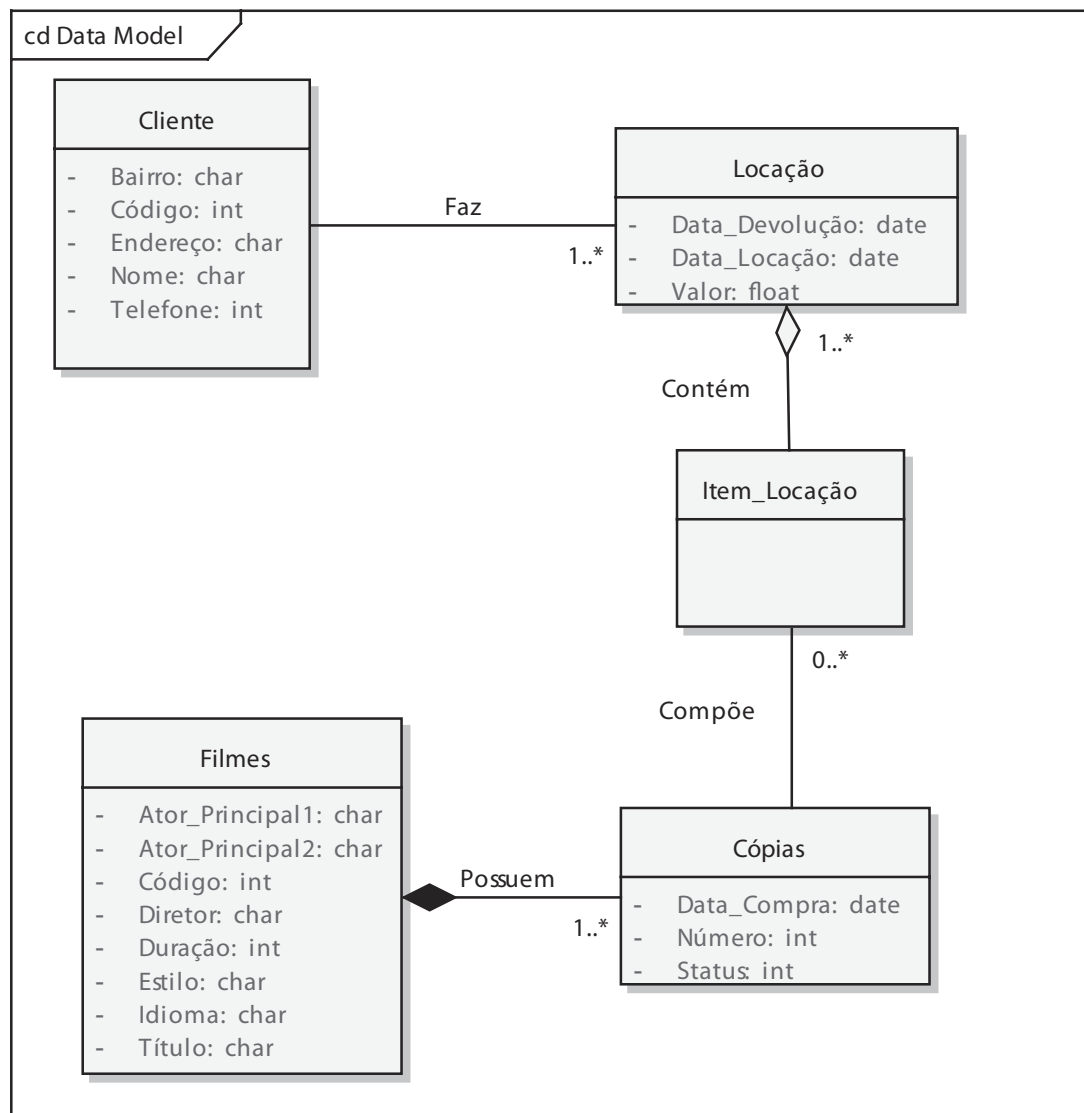



Figura 6.19 – Diagrama de classes sistema videolocadora
Fonte: Elaboração da autora (2008).

Lembra-se do exemplo sobre o sistema bancário?

Imagine uma situação em que você é convidado a desenvolver um projeto para um caixa eletrônico bancário. O projeto prevê o atendimento dos seguintes requisitos funcionais:

- O sistema deve permitir ao cliente a emissão de saldo somente da conta corrente.
- O sistema deve permitir ao cliente a emissão de extrato somente da conta corrente.
- O sistema deve permitir a atualização dos dados cadastrais do cliente.
- O sistema deve permitir o saque em dinheiro no caixa eletrônico.
- O sistema deve permitir a consulta a toda a movimentação financeira do cliente (conta corrente, poupança e aplicações) no caixa eletrônico.
- O acesso às funcionalidades do sistema deve ser possível somente após a verificação da conta e senha do cliente ou gerente.

Observe o possível diagrama de classes para esse exemplo:

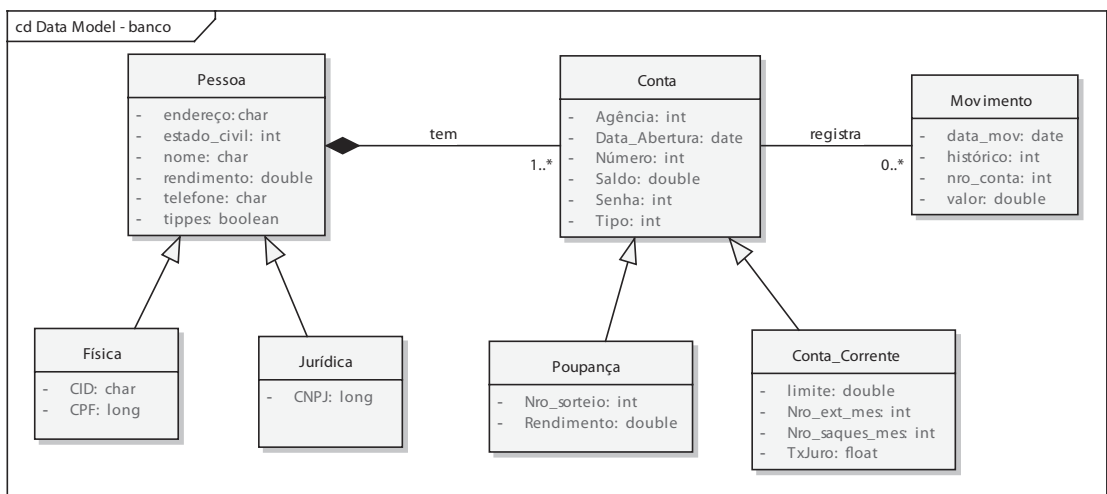



Figura 6.20 – Diagrama de classes sistema caixa eletrônico
Fonte: Elaboração da autora (2008).

Seção 4 – Como ocorre a divisão das classes do modelo de análise?

A divisão, ou categorização, das classes está fortemente relacionada com as futuras mudanças no sistema.

Com a divisão das classes a partir de suas responsabilidades, no momento em que forem necessárias alterações no sistema, estas passam a ser pontuais.

As classes do modelo de análise podem ser divididas em três camadas:

a) classes de Fronteira;

b) classes de Entidade;

c) classes de Controle.

Conheça a partir de agora as características de cada uma dessas camadas de classes.

a) Classes de fronteira – Tratam da comunicação com o ambiente do produto, modelando as interfaces do produto com usuários e outros sistemas (entrada e saída de dados).

Cada formulário usado pelo programa é um objeto criado por uma classe de fronteira, que faz a interface entre um ator e o sistema, uma para cada formulário, relatório ou interface com outro sistema.

Segundo Bezerra (2002), as classes de fronteira têm tipicamente as seguintes responsabilidades:

- notificar as classes de controle sobre eventos gerados externamente ao sistema;
- notificar atores do resultado da interação entre os objetos internos.

São alguns exemplos de classes de fronteira: Formulário Cadastro Cliente, Formulário Cadastro DVDs e Formulário Movimentação DVDs.

b) Classes de entidade – Modelam informações persistentes do sistema, normalmente independentes da aplicação ou das entidades do mundo real.

As classes de entidade criam objetos que gerenciam dados. Assim, você pode vê-los de forma correspondente ao banco de dados. Um ator não tem acesso a uma classe Entidade e a comunicação ocorre por meio de outros objetos.

Exemplos de classes de entidade: Cliente, Filmes, Locação, Cópias.

As classes de entidades armazenam as informações mantidas pelo sistema. Também é importante para o projeto uma definição da performance esperada no acesso aos objetos da classe.

Lembra-se do estudo de caso apresentado no artigo *A importância de utilizar UML para modelar sistemas: estudo de caso*, estudado na unidade 5? Ele se encontra na midiateca. Esse estudo discorre sobre um sistema de vendas de CDs musicais pela internet. A figura 6.21 mostra as classes persistentes encontradas para esse projeto.

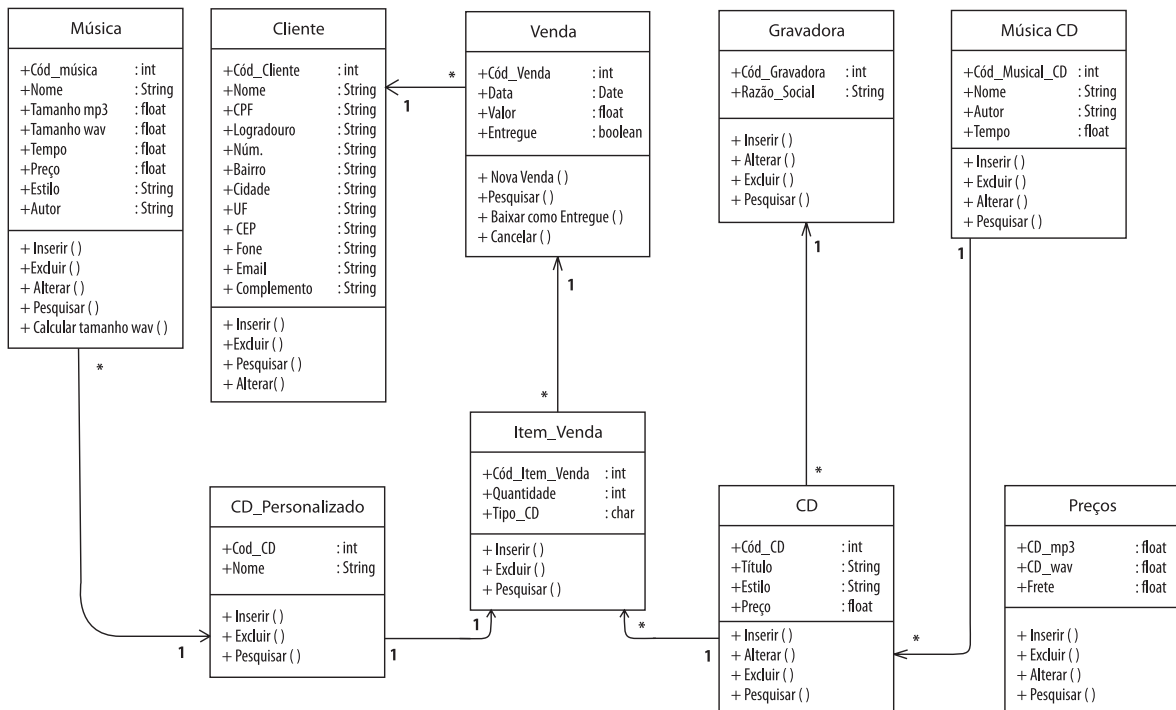



Figura 6.21 – Diagrama de classes Persistente
Fonte: Figueira (2005).

c) Classes de controle – Objetos de controle são pontes de comunicação entre objetos de fronteira e objetos de entidade.

Essas **classes** são responsáveis por controlar a lógica de execução correspondente ao caso de uso. Você pode dizer que elas representam a lógica do caso de uso, requisitam e consultam informações das classes de entidade e de interface e não gerenciam dados nem têm saída visível.

As classes de controle atuam entre as classes de interface e as de negócio, isto é, uma para cada caso de uso.

Bezerra (2002) define algumas responsabilidades para as classes de controle:

- realizar monitorações respondendo a eventos gerados pelas classes de fronteira;
- coordenar a realização do caso de uso por meio de mensagens das classes de entidade e de fronteira;
- assegurar que as regras de negócio do caso de uso estão sendo seguidas;
- coordenar a criação de associações entre classes de Entidade.

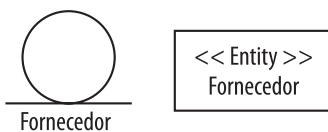
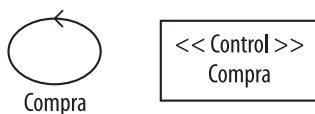
São exemplos de classes de controle: Controlador Cliente, Controlador Entrada DVDs, Controlador Saída DVDs, Controlador Atrasos.

O uso da classe de controle é opcional em um sistema. Você deve avaliar claramente. O objetivo da classe de controle é comportar a lógica e as regras de negócio complexas. Isso significa que ações como inclusão, alteração, consultas de cadastros podem tranquilamente ser implementadas em uma classe de fronteira.

Lembre-se de que os casos de uso complexo devem ser escritos em classes de Controle.

Você se lembra do *software* de declaração do imposto de renda disponibilizado pelo governo federal? As interfaces do sistema com o usuário podem ser descritas pelas classes de Fronteira (tela de cadastro, de registro de bens, entre outras), os dados existentes sobre o trabalhador, rendas, despesas e bens são descritos pelas classes de entidade e o cálculo do imposto de renda será descrito na classe de controle.

A representação das classes na UML se dá pela seguinte notação:

CLASSES DE ENTIDADE

CLASSES DE CONTROLE

CLASSES DE FRONTEIRA

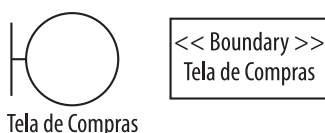



Figura 6.22 – Estereótipos da classe
Fonte: Elaboração da autora (2008).

Você deve estar se perguntando: por que uma classe de entidade não deve cuidar de aspectos relacionados às entradas e saídas? Por que é sugerido o uso de uma classe de fronteira?

Imagine que no sistema de videolocadora a aplicação foi desenvolvida para desktop, mas agora o cliente deseja que o software rode na internet (o que significa uma interface bastante diferente). Bem, se o projeto foi constituído considerando as três classes de Domínio, a equipe de desenvolvimento deve apenas reconstruir as classes de Fronteira, na qual temos as telas do sistema. Isso contribui para a eficiência da manutenção do produto.

Para o sistema de videolocadora, você pode ter algumas classes candidatas:

- classes de entidade: Filme, Cliente, Locação, Cópias;
- classes de fronteira: interface para o cadastro do Cliente (Form_Cliente), interface para o cadastro do Filme (Form_Filme), interface para o movimento da locação (Form_Locação) são alguns exemplos;
- classes de controle: o cadastro do cliente pode ter uma classe de controle para implementação de métodos (Ctrl _Cliente) assim como o movimento da locação (Ctrl_ Locação).

O que fazer então depois de realizada a etapa de identificação das classes?

Finalizada a etapa de identificação das classes de controle, fronteira e entidades, você pode organizar essas classes em **pacotes lógicos**.

Pacotes lógicos são agrupamentos de elementos de um modelo. O uso de pacotes agrupa classes que possuem um critério comum, facilitando a comunicação.

Quando uma coleção de classes colabora entre si para realizar um conjunto coeso de responsabilidades, ela pode ser vista como um subsistema.

De acordo com Pressman (2002), quando visto de fora, um subsistema pode ser tratado como uma caixa-preta que contém um conjunto de responsabilidades e suas próprias colaborações.

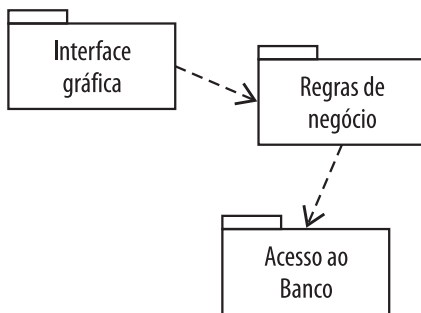



Figura 6.23 – Pacotes lógicos
Fonte: Elaboração da autora (2008).

É importante ressaltar que o pacote deve ter um nome único e textual.

Seção 5 – O que é diagrama de objetos?

Os diagramas de objetos são como uma fotografia de um sistema orientado a objetos em execução.

O diagrama de objetos é um complemento do diagrama de classes, pois fornece uma visão dos valores armazenados pelos objetos de um diagrama de classes em um determinado momento da execução de um processo de software (GUEDES, 2006).

Em alguns livros, você vai encontrar o nome "diagrama de instâncias" como sinônimo de "diagrama de objetos".

Quando fazer uso desse **diagrama**? Esse diagrama pode ser bastante útil quando você estiver modelando uma estrutura de dados complexa.

O diagrama de objetos é representado por um retângulo com dois compartimentos. Na parte superior, você identifica o objeto (sublinhado). Na parte inferior, você referencia os atributos com seus valores.

Analise a figura a seguir. Note que na parte superior estão as três classes associadas: Pedido, itemPedido e Produtos. A instância Pedido está associada a duas instâncias do itemPedido, que consequentemente está ligada a uma instância do Produto.

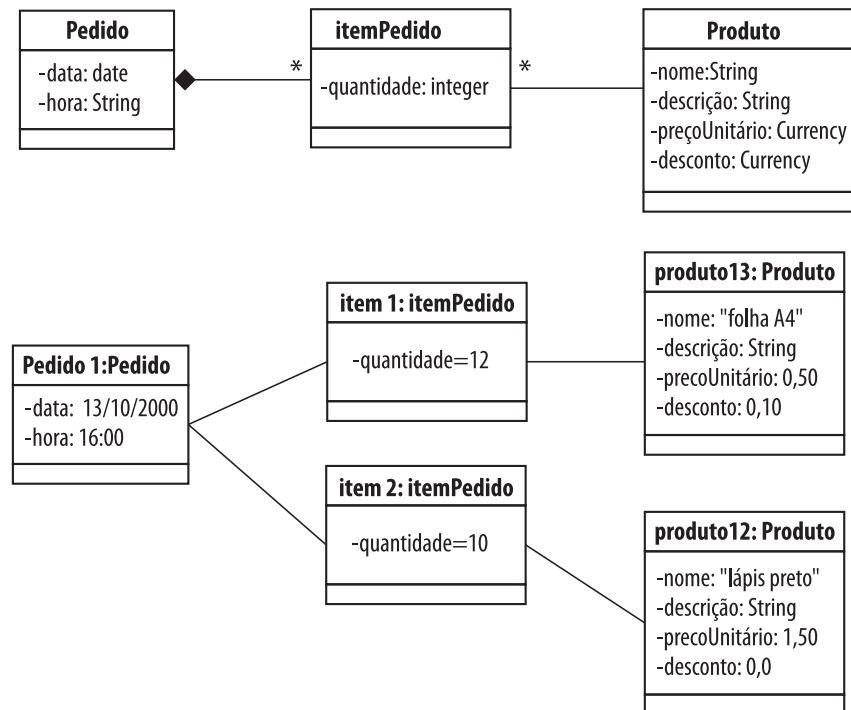



Figura 6.24 – Diagrama de objetos
Fonte: Adaptado de Bezerra (2000).

Qual a nomenclatura a ser utilizada na especificação de um diagrama de objetos?

Existem duas possibilidades: os atributos e as operações.

a) Atributos – Os atributos foram apresentados até o momento utilizando-se apenas o nome, mas com certeza em seu projeto você terá de explicitá-los de forma mais detalhada.

A sintaxe a ser apresentada é:

[1] visibilidade nome:tipo=valor inicial

A visibilidade refere-se ao nível de acesso: quantos atributos de um objeto estarão visíveis a outros objetos. Pode ser:

- + Pública – Todos têm acesso, podendo ser utilizado por operações declaradas dentro de outras classes.
- # Protegida – Pode ser acessado apenas por operações dentro da própria classe, pelas classes da hierarquia e pelas classes do "pacote".
- – Privada – Só pode ser acessado por operações dentro da própria classe.

O uso das propriedades de visibilidade apoia um dos conceitos da orientação a objetos: o **encapsulamento**. Assim, você só deixa visível atributos realmente necessários aos demais objetos enquanto outros atributos tornam-se invisíveis.

Pedido
+ código: int #data: Data -total: float -desconto: float #dataEntrega:Data

Figura 6.25 – Nomenclatura de atributos
Fonte: Elaboração da autora (2008).

Sobre a figura 6.25, acompanhe:

- **Nome** – Identificador do atributo.
- **Tipo** – O tipo depende da linguagem de programação. Mas é comum o uso de uma tipologia abstrata em que são definidos os tipos como inteiro, real, caractere, *string*, *float*, *data* ou outra classe.
- **Valor Inicial** – Você pode informar um valor inicial para um atributo. Quando o objeto da classe for instanciado, ele pegará o valor automaticamente (veja limiteCredito).

b) Operações – As operações representam o conjunto de funcionalidades da classe. Para cada operação, especifica-se sua assinatura:

- **Nome** – Identificador para o método.
- **Tipo** – Quando o método tem um valor de retorno, o tipo desse valor.
- **Lista de argumentos** – Quando o método recebe parâmetros para sua execução, o tipo e um identificador para cada parâmetro.
- ***In*** – parâmetro de entrada; *out* para um parâmetro de saída; e *inout* para parâmetros de entrada que podem ser modificados.
- **Visibilidade** – Utiliza-se dos mesmos recursos usados para os atributos, definindo o quão visível é uma operação a partir de objetos de outras classes.

Pedido
+obterProduto():String +obterLimite():Moeda +obterMediasComprar():float

Figura 6.26 – Exemplo de operação
Fonte: Elaboração da autora (2008).

Síntese

Agora que você já estudou a modelagem de classes, aproveite para praticar os conhecimentos conquistados nesta unidade realizando as atividades propostas a seguir.

Nesta unidade, você aprendeu que um objeto é algo que tem estado, comportamento e identidade. Uma classe é uma definição abstrata de um conjunto de objetos que compartilham uma estrutura e um comportamento comuns – todo sistema, porém, engloba muitos objetos que cooperam entre si para produzir a funcionalidade desejada.

A produção das funcionalidades só é possível pela existência de diferentes tipos de relacionamentos, como a associação, a dependência e a generalização.

Entre as características do relacionamento de associação, a multiplicidade é uma das mais importantes. A multiplicidade indica o número de instâncias que participam dessa associação. Você também viu que as operações determinam como um objeto age e reage às mensagens que ele recebe e que é possível agrupar esses objetos e classes em pacotes lógicos, criando uma visão mais clara do sistema.

Você também estudou a importância de modelar o sistema em diferentes camadas, como as camadas de Controle, Fronteira e Persistente. Essa modelagem cuidadosa facilita futuras manutenções no seu projeto.

Em resumo, esta unidade englobou conceitos e abstrações relacionadas aos aspectos estáticos do sistema. Mas, para o bom andamento do projeto, é necessário ter uma visão dinâmica desses objetos.

Atividades de autoavaliação

Leia com atenção os enunciados e realize as questões propostas.

1) Assinale a afirmativa correta.

a) ( ) Uma classe é um conjunto de objetos, que, por sua vez, são identificados por comportamento e estado e nem sempre são únicos.

b) ( ) Uma classe é um conjunto de objetos que compartilham características de atributos, operações, relações e semântica.

c) ( ) É possível dizer que são exemplos de classes em um sistema Universitário: Professor, Aluno, Nome Aluno.

d) ( ) É possível dizer que são exemplos de instâncias de uma classe Professor: "João da Silva", "Ana Luiza".

2) Relacione a primeira coluna com a segunda, indicando a camada mais adequada para cada ocorrência.

A. Classe de Ccontrole

B. Classe Persistente

C. Classe de Fronteira

( ) Mensagens de erro para o usuário.

( ) Cálculo da folha de pagamento.

( ) Cria ou destrói um objeto (exclui produto).

( ) Dados do produto.

( ) Telas de consulta de produtos.

( ) Dados do cliente.

3) Relacione a primeira coluna com a segunda, indicando as características dos diferentes relacionamentos:

A. Associação

B. Associação ternária

C. Agregação

D. Herança

E. Dependência

F. Associação

recursiva

G. Classes associativas

a) ( ) As classes estão ligadas a associações e não a outras classes.

b) ( ) Neste relacionamento, ocorre a associação de três classes ao mesmo tempo.

c) ( ) Uma das classes do relacionamento é uma parte da classe ou está contida em outra classe.

d) ( ) Este relacionamento é possível entre dois ou mais elementos em que uma classe Cliente é dependente de algum serviço da classe Fornecedora.

e) ( ) Neste caso, os objetos da própria classe estão se relacionando.

f) ( ) Quando ocorre este tipo de relacionamento, a subclasse herda as propriedades da superclasse, principalmente atributos e operações.

g) ( ) Ocorre quando duas classes ou mesmo uma classe, consigo própria, apresenta interdependência.

4) Com base no texto a seguir relacionado à Clínica "Bem-Estar":

a) Identifique as classes persistentes (nome e descrição da classe):

b) A partir dessa identificação, construa o diagrama de classes Persistentes, apontando os relacionamentos existentes entre as classes.

Empresa : Clínica Bem-Estar

1) Função: fomos contratados para analisar seu processo atual e verificar como expandir suas operações e melhorar seu nível de serviço.

Histórico:

A clínica, fundada há 5 anos, atua no atendimento clínico pediátrico.

A clínica possui 34 médicos cadastrados em diferentes especialidades como: cardiologia, clínica geral, dermatologia etc. Todos os médicos utilizam internet e e-mail. A faixa etária predominante é de 30, 35, 40, 42, 44 e 48 anos. Todos os médicos são aptos do ponto de vista físico.

O paciente pode ser atendido de forma particular ou por convênios. Os convênios atendidos são o Bruxtr, Vpfzm e UIOlk.

Cada médico faz 3 plantões semanais de 4 horas seguidas; as consultas possuem um intervalo de 30 minutos. Existe a possibilidade de a consulta ser de retorno, nesse caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde são marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio. Trabalham há 3 anos na clínica com planilhas Excel.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente, que contém nome, endereço, telefone, data de nascimento, convênio.

O médico, ao atender o paciente, preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

A clínica possui de 700 a 800 fichas, sendo que cerca de 600 são de atendimento por convênio.

O gerente da clínica está ansioso, pois não consegue controlar questões relacionadas ao número de pacientes atendidos por convênio e particular, médicos mais procurados e picos de movimento.

Volume de atendimentos: 56 por dia.

Outra questão de interesse é manter um controle de laboratórios conveniados, pois o médico poderia indicar o laboratório já no momento da prescrição.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar:

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. São Paulo: Campus, 2002. (Ler capítulos 5, 6 e 8.)

BOOCH, G. ; RUMBAUGH, J. ; JACOBSON, I. **UML:** guia do usuário. São Paulo: Campus, 2000. (Ler capítulos 8 e 9.)

PAULA FILHO, Wilson de Pádua. **Engenharia de *software*.** São Paulo: Campus, 2001. (Ler capítulo 4.)

Você encontra uma boa leitura sobre modelagem de chaves no capítulo 4 do livro **UML: uma abordagem prática**, de Gilleanes T. A. Guedes.

UNIDADE 7

Modelos de interações

Objetivos de aprendizagem

- Entender os elementos existentes no modelo de interação oferecido pela UML.
- Compreender as características existentes entre as diferentes mensagens utilizadas na comunicação entre objetos.
- Perceber quando o uso de diagramas de interação pode ser interessante para a compreensão de um projeto de software.

Seções de estudo

Para início de estudo

A modelagem de casos de uso identifica o que o sistema deve fazer e para quem deve ser feito. Apesar de fazer essa descrição, não são especificados mais detalhes relacionados ao comportamento interno do sistema na execução das funcionalidades.

O modelo de classes de domínio estudado na unidade 6 agrega ao projeto a visão estática e estrutural do sistema. Sob essa visão, você construiu a definição das classes e responsabilidades. Apesar de você já ter até aqui uma ideia clara do sistema com esses dois modelos, nenhum aspecto relacionado ao mecanismo de comunicação e comportamento entre objetos foi tratado até este momento.

Por isso, nesta unidade, você vai estudar sobre o modelo de interações, que apresenta as mensagens trocadas entre os objetos na execução de um determinado cenário.

O uso do modelo de interações procura descrever o modelo dinâmico do sistema propondo a descrição, inclusive temporal, da troca de mensagens entre objetos.

Seção 1 – Quais são os elementos do modelo de interação?

Segundo Booch (2000), interação é como um comportamento que abrange um conjunto de mensagens trocadas entre um conjunto de objetos em um determinado contexto para a realização de um propósito.

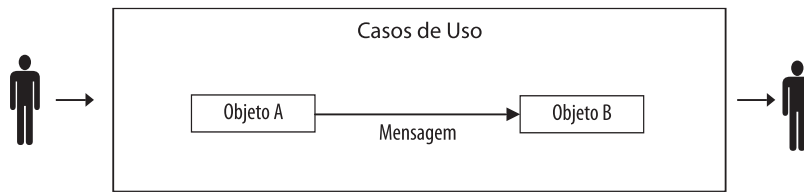



Figura 7.1 – Interação entre objetos
Fonte: Elaboração da autora (2008).

A interação é empregada para a modelagem do fluxo de controle de uma operação, uma classe, um componente, um caso de uso ou do sistema como um todo.

O uso de interações também introduz mensagens que são enviadas de objeto a objeto. Essas mensagens envolvem a chamada a uma operação ou o envio de um sinal.

No decorrer do seu estudo, você já leu várias vezes a palavra "mensagem", certo? É provável que você já tenha uma ideia sobre o significado dessa palavra, e, por isso, nessas três últimas unidades parecia irrelevante conceituar esse substantivo. Mas, nesta unidade, essa palavra se torna o elemento fundamental do modelo.

Segundo Bezerra (2002), uma mensagem é uma solicitação de execução de uma operação em outro objeto. Um objeto pode ainda enviar uma mensagem para si mesmo (mensagem reflexiva).

O uso de uma mensagem em um diagrama de interação permite a passagem de informações que são repassadas para a operação, que será executada no objeto receptor.

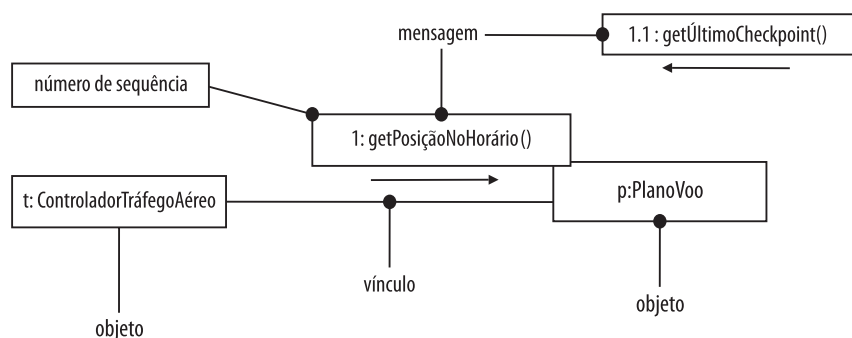



Figura 7.2 – Interação entre objetos de operação de voo
Fonte: Elaboração da autora (2008).

Na figura 7.2, são considerados objetos:

t:ControladorTráfegoAéreo e p:PlanoVoo;

1: getPosiçãoNoHorário() e 1.1 : getÚltimoCheckpoint() são consideradas mensagens.

Os números 1 e 1.1 nas mensagens são números de sequência usados para organizar a sequencialização das mensagens.

As mensagens são representadas graficamente por linhas com uma direção e quase sempre incluem os nomes de suas operações, os objetos e seus vínculos, como ilustra a figura 7.2, na qual a troca de mensagens ocorre entre os objetos Plano Voo e Controlador Tráfego Aéreo.

Os **objetos** de uma interação desempenham determinados papéis, como você pode perceber na figura 7.3. Nessa figura, a classe Pessoa representa o papel do empregado e a classe Empresa, o papel do empregador.

Já os **vínculos** constituem normalmente uma instância de uma associação. Um vínculo especifica um caminho pelo qual um objeto pode enviar uma mensagem para outro objeto ou para o mesmo objeto (BOOCH, 2000). Observe a figura a seguir:

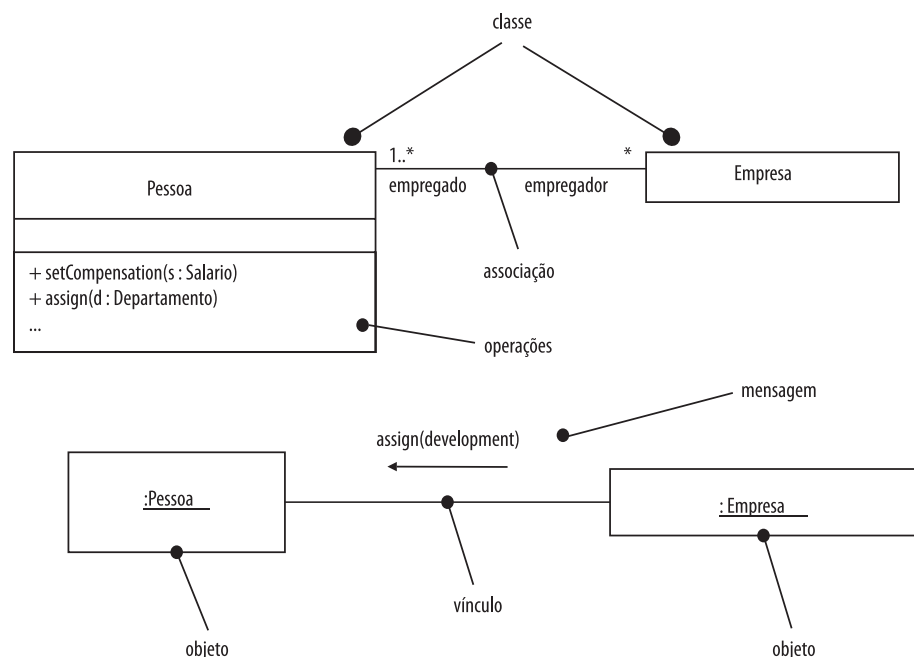
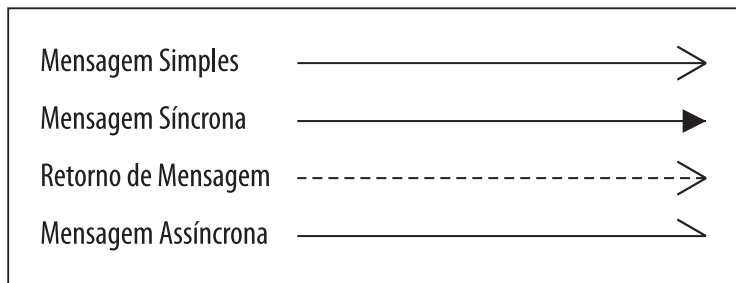



Figura 7.3 – Interação entre objetos
Fonte: Booch (2000).

Quais são os tipos de mensagem?

A notação UML descreve a possibilidade de três tipos de mensagens (BEZERRA, 2002):

- **Mensagem simples** – Este tipo de mensagem é utilizado quando a natureza da mensagem é irrelevante.
- **Mensagem síncrona** – Indica que o objeto remetente espera que o objeto receptor processe a mensagem antes de recomeçar o seu processamento. Neste caso, o objeto receptor ficará bloqueado até que a requisição seja atendida.
- **Mensagem assíncrona** – O objeto remetente não espera resposta para prosseguir seu processamento.

A mensagem é representada por uma seta em que o sentido é do objeto remetente para o objeto receptor. As mensagens possuem um rótulo que procura especificar as informações que devem transmitir.

Você pode usar a seguinte sintaxe:

```
Expressão-sequência recorrência: v := mensagem
```

Assim, tem-se:

a) **Expressão-sequência** – As mensagens são passadas de um objeto para outro. Esse fluxo de mensagens forma uma sequência. As sequências devem ter um ponto de partida indicando o início do processo. A expressão de sequência elimina ambiguidades acerca de quando a mensagem foi enviada em relação às demais.

1: AtenderChamado()

Sequência é o número 1, a expressão AtenderChamado()

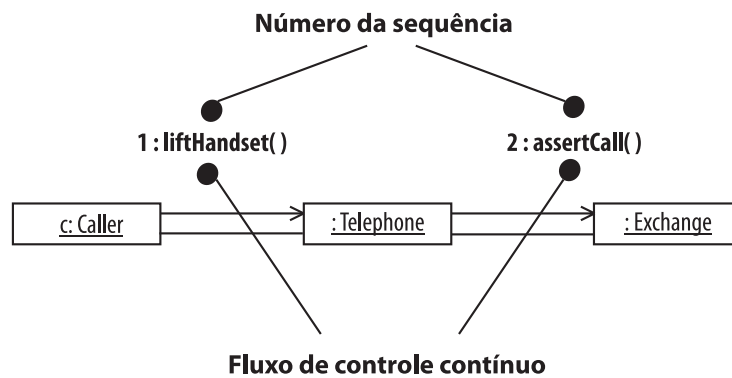



Figura 7.4 – Sequenciamento das mensagens
Fonte: Booch (2000).

Na figura 7.4, observe a numeração das mensagens (1 e 2), que indica a direção e a ordem em que elas acontecem.

b) **Recorrência** – Às vezes o envio de uma mensagem está condicionado ao valor de uma expressão lógica (verdadeiro ou falso) ou ao número de vezes que a mensagem será enviada. Se a recorrência for uma cláusula-condição, então a mensagem será enviada somente se a condição for verdadeira.

Sua sintaxe é sempre entre colchetes: [cláusula-condição]

[existe produto estoque] EfetuarVenda()

A repetição é ordenada pelo uso de asterisco:
*[cláusula iteração]

*[enquanto Número_Itens < 10] InserirItem()

c) **Variável** – Identifica uma variável que recebe o valor de retorno da operação executada pelo receptor.

1.2.1: Z :=verificarEstoque(e)

A variável Z vai receber o retorno da operação verificarEstoque.

Quando uma mensagem é enviada, você está especificando uma comunicação entre objetos, que possuem uma expectativa de realização de uma atividade. Quando a mensagem é passada, o resultado é uma ação na forma de uma instrução executável.

Você pode fazer a modelagem de vários tipos de ação, como:

- **Call** (mensagens síncronas) – Solicita uma operação em um objeto.
- **Send** (mensagens assíncronas) – Envia um sinal para um objeto.
- **Create** – Cria um objeto.
- **Destroy** – Destrói um objeto.
- **Return** (retorno) – Retorna o controle a quem ativou um *Call*.

Seção 2 – O que é Diagrama de Interação?

Um diagrama de interação mostra uma interação formada por um conjunto de objetos e seus relacionamentos, incluindo as mensagens que poderão ser trocadas entre eles.

O diagrama mostra como devem ser implementadas as ações focalizando os objetos que devem ser criados para a implementação da funcionalidade requerida no caso de uso (LIMA, 2005).

Um diagrama de interação pode modelar um caso de uso, assim como pode ser necessário o uso de vários diagramas para modelar a interação de um caso de uso que possui diferentes cenários.

Existem dois diagramas de interação: o diagrama de sequência e o diagrama de comunicação.

a) Diagrama de sequência

O **diagrama de sequência** mostra a sequência de eventos que ocorrem em um determinado processo, e apresenta quais condições devem ser satisfeitas e quais métodos devem ocorrer entre os objetos envolvidos e sua ordem durante a execução do processo (GUEDES, 2006).

Para construir um diagrama de sequência, é necessária a prévia definição do diagrama de classes com a indicação das operações associadas.

A descrição é sempre uma interação dentro de uma unidade de tempo. É ideal para a especifi-cação de processos que ocorrem em tempo real.

O diagrama de sequência descreve o comportamento interno, mostrando os eventos entre objetos, mas omite a associação entre esses objetos.

A notação usada pela UML para representar o diagrama de sequência utiliza-se de atores, objetos, classes e mensagens, conforme mostra a figura a seguir.

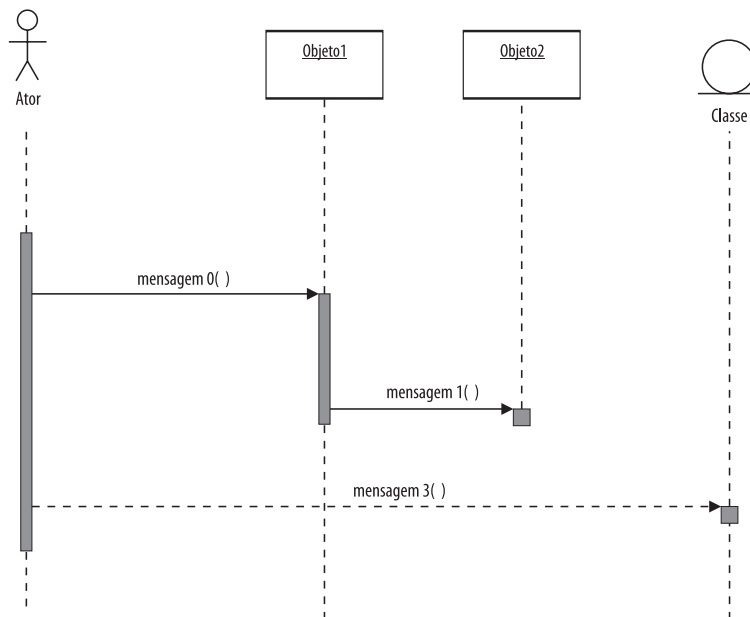



Figura 7.5 – Elementos gráficos do diagrama de sequência
Fonte: Elaboração da autora (2008).

- **Atores** – Participam do diagrama de sequência opcionalmente, dependendo do cenário do caso de uso.
- **Objetos** – A ordem na qual os objetos aparecem não é predefinida, mas normalmente utiliza-se a ordem da esquerda para a direita: ator, objetos de fronteira, objetos de controle, objetos de entidade e atores secundários.
- **Classes** – Aparece no diagrama quando uma mensagem for endereçada para a classe e não para o objeto.
- **Linhas de vida** – Cada objeto aparece no topo de uma linha vertical tracejada, é a linha de vida.
- **Mensagem** – São as linhas horizontais com flechas que ligam uma linha de vida a outra. As flechas horizontais são rotuladas com as mensagens.

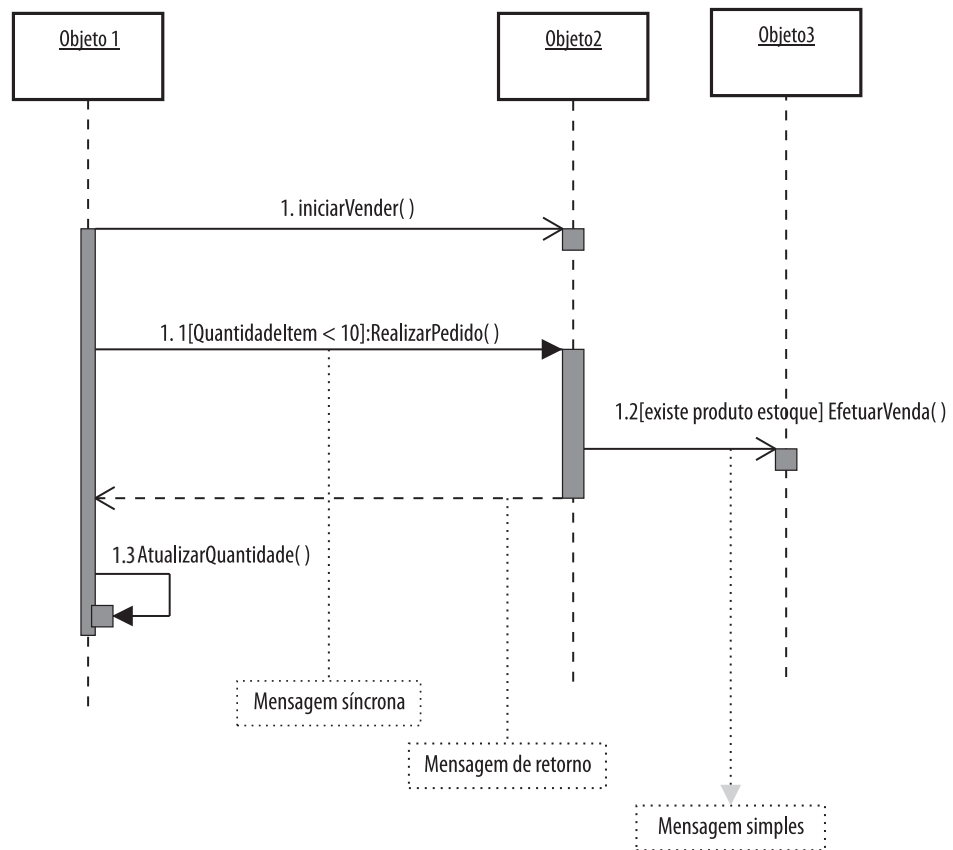



Figura 7.6 – Diagrama de sequência
Fonte: Elaboração da autora (2008).

- **Focos de Controle** – Os focos de controle são as caixas retangulares que estão sobre a linha de vida do objeto. O foco de controle indica o tempo necessário para que o objeto realize uma ação. O início do foco deve estar na altura da flecha de mensagem. O final deve coincidir com o final da atividade realizada pelo objeto.

A figura 7.7 mostra o diagrama de sequência de um caso de uso para registro de uma venda. O ator Atendente envia uma mensagem para totalizar o objeto Venda. Em seguida, o ator dispara a mensagem para registrar o modo de venda para o objeto Venda.

A partir de então, uma recorrência condicional é estabelecida: se a venda for a prazo, envia a mensagem *Inserir*, sendo o parâmetro o próprio objeto Venda para o objeto Contasareceber. Se a venda for à vista, envia a mensagem *registrapagamento* para o objeto Caixa.

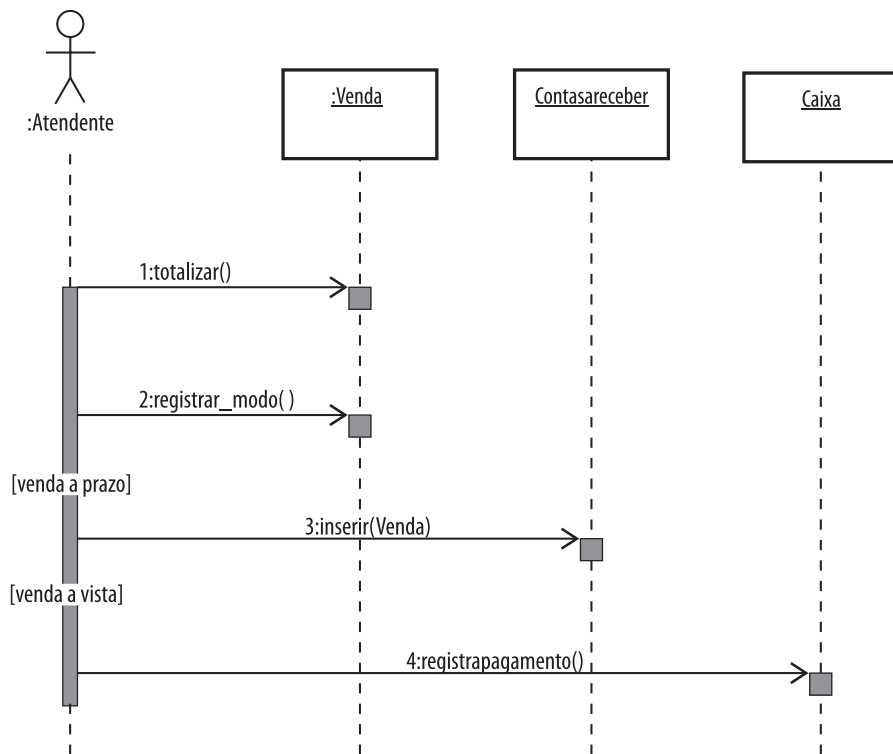



Figura 7.7 - Diagrama de sequência
Fonte: Pádua (2001).

Agora observe a figura a seguir:

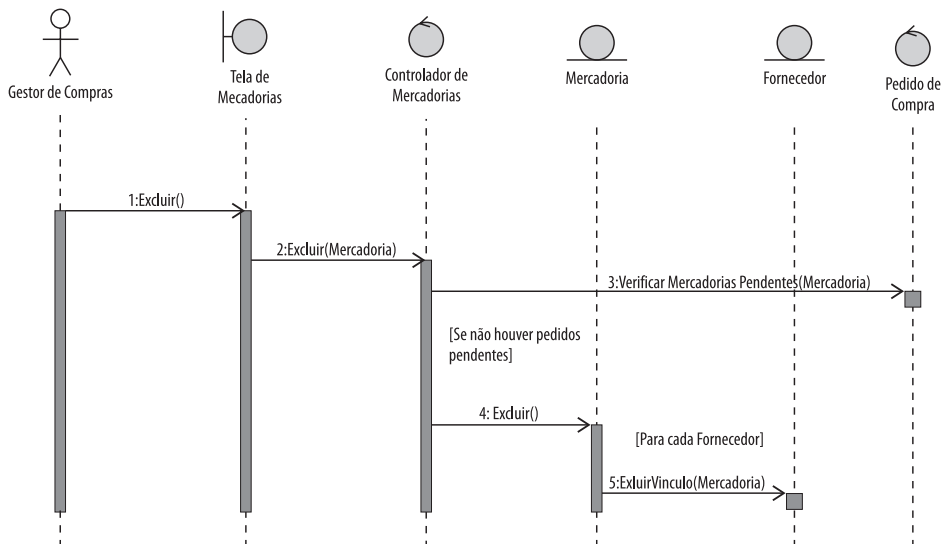



Figura 7.8 – Diagrama de sequência
Fonte: Pádua (2001).

No diagrama da figura 7.8, são utilizadas classes de Fronteira (Tela de Mercadorias), classes de Controle (Controlador de Mercadorias e Pedido de Compra) e classes Persistentes (Mercadoria e Fornecedor).

Todas as mensagens estão sequenciadas (1-5) indicando a ordenação temporal das mensagens.

Lembre-se de que o diagrama de sequência está baseado na descrição do caso de uso, então ele é um reflexo do que foi documentado. Observe o diagrama de sequência para o sistema de videolocadora para o caso de uso Gerenciar Locações. Perceba que foram usadas apenas as classes de Fronteira e de Entidade.

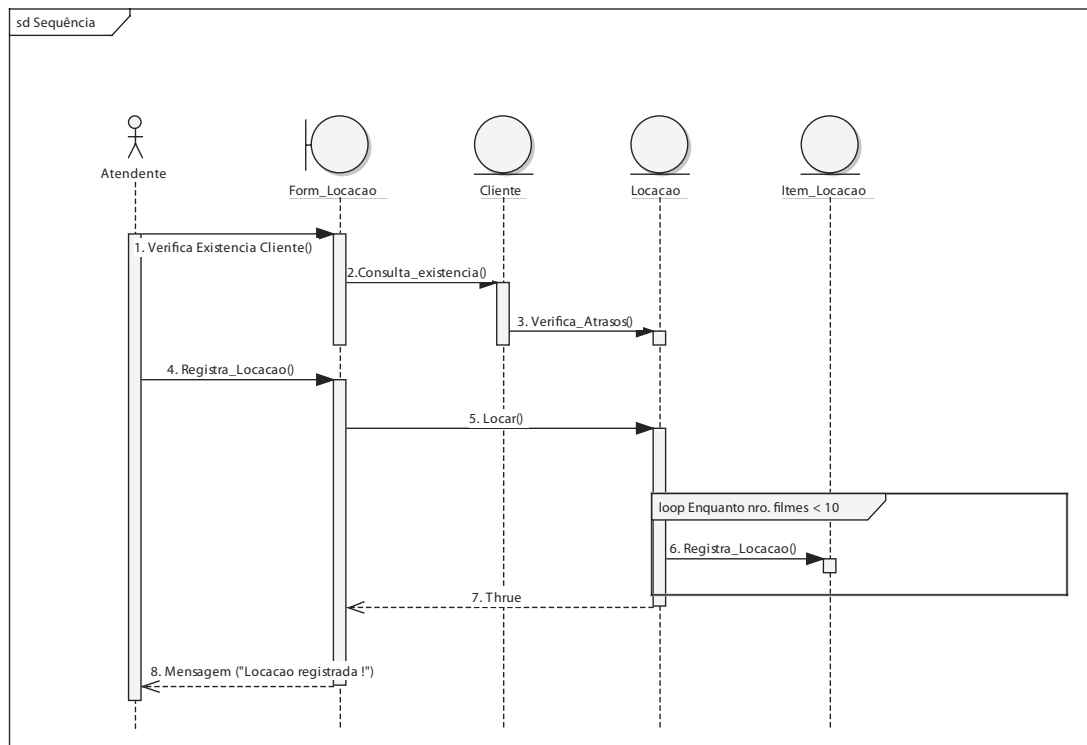



Figura 7.9 – Diagrama de sequência para o sistema de videolocadora
Fonte: Elaboração da autora (2008).

b) Diagrama de colaboração

O diagrama de colaboração é um modo alternativo para representar a troca de mensagens entre um conjunto de objetos. O diagrama de colaboração sempre mostra os objetos relevantes para a execução do caso de uso (GUEDES, 2006).

Nesse diagrama, a ordem em que as mensagens foram enviadas não é apresentada, pois não existe a dimensão de tempo (linhas de vida do diagrama de sequência), o que obrigatoriamente tem-se expressões de sequência em todas as mensagens (1, 1.1, 1.2...).

Diagramas de colaboração apresentam ênfase no sistema, ou seja, são usados para obter uma visão geral do sistema:

- os objetos que são criados durante uma colaboração são especificados como {**new**};
- os que são destruídos durante uma colaboração são especificados com {**destroyed**}; e
- os que são criados e destruídos na mesma colaboração são especificados com {**transient**}.

A leitura do diagrama ou a sequência das mensagens é organizada pelas setas no rótulo da mensagem.

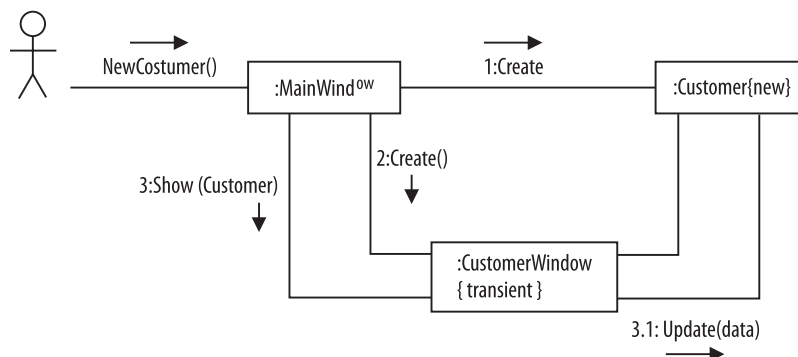



Figura 7.10 – Exemplo de um diagrama
Fonte: Adaptado de Ambler (1998).

O objeto MainWindow recebe a mensagem NewCustomer e cria um objeto Customer. Um CustomerWindow é criado e o objeto Customer é então passado para o CustomerWindow, o qual atualiza os dados do Customer.

No exemplo a seguir, você verá o diagrama de colaboração de um sistema de empréstimo para uma biblioteca.

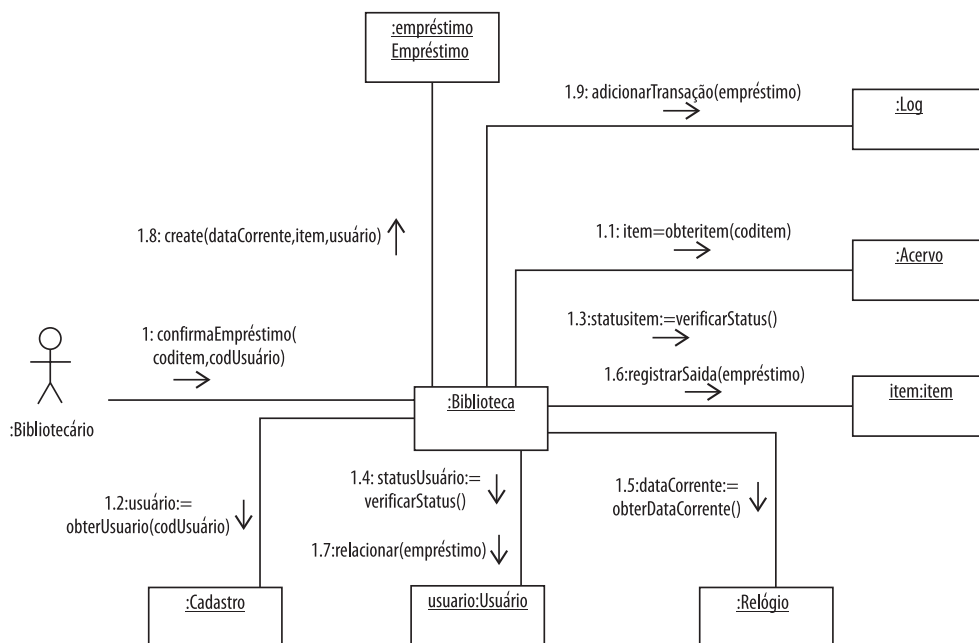



Figura 7.11 – Diagrama de comunicação
Fonte: Liesenberg (2005).

O diagrama inicia a comunicação pela mensagem confirmarEmpréstimo e finaliza-se pela mensagem adicionarTransiçao no objeto Log. Observe que o diagrama apresenta a relação existente entre os diferentes objetos de forma bastante legível pelo uso da numeração e do direcionamento das mensagens por meio de setas.

Afinal, diagrama de sequência ou diagrama de colaboração?

Um diagrama de sequência de sistema representa uma sucessão de eventos de entrada, gerados por um ator ao executar um fluxo de um caso de uso.

Nos diagramas de sequência, existe uma linha de vida do objeto, que é a linha tracejada vertical que representa a existência de um objeto em um período de tempo.

Além disso, existe o foco de controle que mostra o período durante o qual um objeto está desempenhando uma ação, diretamente ou por meio de um procedimento subordinado.

Esse diagrama é interessante na descrição de uma sequência particular de funcionamento, mas pode ser confuso quando existem muitas sequências alternativas.

Os diagramas de colaboração sempre apresentam o caminho que indica como um objeto está vinculado a outro. Além disso, existe o número de sequência para indicar a ordem temporal de uma mensagem.

Se você precisa de um diagrama que demonstre o fluxo de eventos no decorrer do tempo, então deve utilizar o diagrama de sequência; se a ênfase for o contexto do sistema, a melhor opção é o diagrama de colaboração.

Quer conhecer mais?

Para conhecer um pouco mais sobre os modelos de interação, acesse a Midiateca. O texto "Exemplo Sequencia&Colaboração" apresenta o diagrama de colaboração e de sequência para um sistema de videolocadora.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas.

Síntese

Nesta unidade, você foi apresentado ao modelo dinâmico do projeto. Foi possível perceber que essa visão aborda aspectos internos do sistema, chegando ao nível das funções que serão implementadas futuramente.

O uso de diagramas de interação permite visualização e entendimento do funcionamento temporal da troca de mensagens entre os diversos objetos. Você pode representar as mensagens por

meio de diferentes notações, em que é possível apresentar relações síncronas, assíncronas e de retorno entre os objetos.

O uso da representação temporal da troca das mensagens pode ou não ser representado. Se a dimensão de tempo é fundamental para o entendimento, você pode utilizar o diagrama de sequência. Mas se a dimensão do contexto do sistema é o mais importante, utilize o diagrama de colaboração.

Atividades de autoavaliação

Leia com atenção os enunciados e, em seguida, realize as questões propostas.

1) Relacione os conceitos a seguir. Observe que uma mesma opção pode se repetir.

1. Cláusula condição
2. Mensagem
3. *New*
4. *Destroyed*
5. *Transient*
6. Linhas de vida
7. Cláusula interação
8. *Call*
9. *Send*

a) ( ) São as linhas horizontais com flechas que ligam uma linha de vida a outra.

b) ( ) Objetos são destruídos durante uma colaboração.

c) ( ) Os objetos aparecem no topo de uma linha vertical tracejada.

d) ( ) 1:[preço < 10,00]:Venda aVista (produto).

e) ( ) Utilizado para mensagens síncronas.

f) ( ) Objetos são criados e destruídos durante uma colaboração.

g) ( ) 2:* [lê_codigo].

h) ( ) Utilizado para mensagens assíncronas.

i) ( ) Objetos são criados durante uma colaboração.

2) Construa o diagrama de sequência da Clínica Bem-Estar para o caso de uso Agendar Horário.

3) É correto afirmar sobre interação:

a) ( ) Propõe a troca de mensagens entre atores.

b) ( ) Mensagens são trocadas entre um conjunto de objetos em um determinado contexto para a realização de um propósito.

c) ( ) Pode ser definida como uma solicitação de execução de uma operação em outro objeto.

d) ( ) Na mensagem síncrona, o objeto remetente não espera resposta para prosseguir com seu processamento; já na assíncrona o objeto remetente espera que o objeto receptor processe a mensagem antes de recomeçar o seu processamento.

Saiba mais

Caso você tenha interesse em aprofundar seus estudos sobre os assuntos tratados nesta unidade, consulte:

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. São Paulo: Campus, 2002. (Ler capítulo 7.)

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** guia do usuário. São Paulo: Campus, 2000. (Ler capítulos 18 e 27.)

UNIDADE 8

Modelos de estados

Objetivos de aprendizagem

- Reconhecer objetivos e características existentes na modelagem da visão dinâmica do projeto.
- Compreender a notação utilizada nos diagramas que complementam a especificação do modelo dinâmico do sistema.
- Reconhecer as possibilidades de utilização e as notações envolvidas no diagrama de estados e no diagrama de atividades.

Seções de estudo

Para início de estudo

O modelo dinâmico do sistema completa-se a partir de cinco diagramas: o de atividades, o de sequência, os de colaboração, o de estados e o de casos de uso. Cada um desses diagramas especifica e esclarece aspectos diferentes do sistema.

Até este momento, você estudou três desses diagramas (casos de uso, colaboração e sequência). Nesta unidade, serão apresentados os diagramas de transição de estado que modelam o comportamento de um objeto e o diagrama de atividade que modela a sequência geral de ações para vários objetos e casos de uso.

Imagine um semáforo de rua. É possível prever três estados para ele: verde, amarelo e vermelho. O diagrama que permite especificar essa transição entre os eventos é o diagrama de estados. A mudança de estado dispara ações diferentes do sistema, que, por sua vez, modificam o estado do objeto (sinal vermelho: ação parar!).

Já os diagramas de atividade conseguem especificar situações, como paralelismos e sincronizações, que são impossíveis de serem especificadas no diagrama de casos de uso.

Seção 1 – Modelo de estados

Os objetos de um sistema modificam seu estado de forma análoga a objetos do mundo real. Pense no cozimento do macarrão. Em seu primeiro estado, ele está duro, pois não está cozido. Depois de cozido ele amolecerá e assim entra em outro estado. Essa modificação é chamada de transição entre estados.

Você pode dizer que um estado representa o resultado de atividades executadas por um objeto, determinada pelos valores de seus atributos e pela sua ligação com outros objetos (FURLAN, 1998).

Os objetos do sistema passam por vários estados. Essa transição faz com o próprio sistema se modifique.

Na verdade, **a transição ocorre porque um evento (mensagens, timer, erros, condições sendo satisfeitas, entre outros) disparado no sistema faz com que o objeto realize determinadas ações, que fazem com que o objeto modifique seu estado.**

O diagrama de estados (DTE) deve ser utilizado somente para algumas classes, ou seja, somente para aqueles que possuem estados bem definidos e onde a mudança de estados propicia a mudança do comportamento das classes.

No diagrama a seguir, é possível mapear os possíveis estados dos objetos e as ações e condições necessárias para que ocorra a mudança de estados entre os objetos.

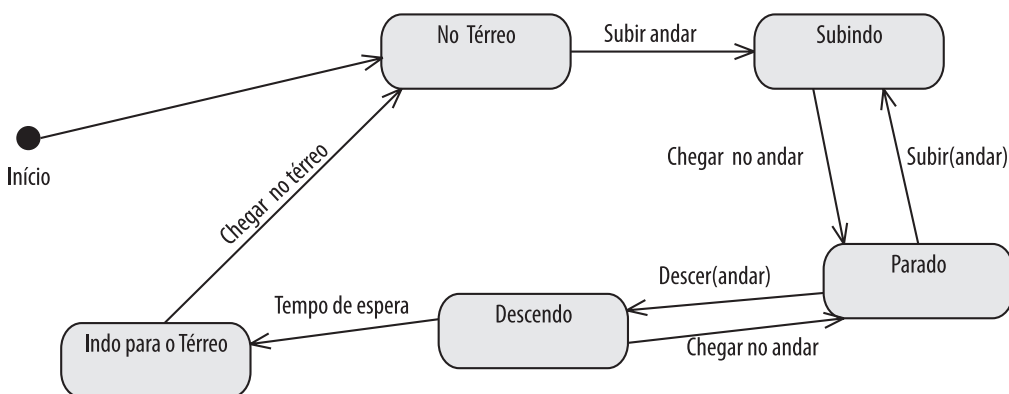



Figura 8.1 – Diagrama de estados elevador
Fonte: Elaboração da autora (2008).

O DTE dessa figura mostra:

- os estados de um objeto;
- os eventos ou as mensagens que causam transição;
- as ações que resultam de uma mudança de estado.

Quais são os componentes de um diagrama de estados?

A existência de estado em um objeto indica a ordem em que as operações são executadas. No DTE, é importante a descrição da ordem das operações no tempo, pois essa ordem pode formalizar a caracterização do comportamento de um objeto.

O DTE utiliza-se de alguns componentes, são eles:

a) Estado – É uma situação na vida de um objeto durante a qual ele satisfaz alguma condição ou realiza alguma atividade em resposta a um evento ou espera a ocorrência de algum evento. No diagrama DTE, o estado é representado por um retângulo arredondado.

Emitindo nota/O macarrão está cozido/O menino está nadando.

Objetos: nota, macarrão, menino.

Estado: emitindo, cozido, nadando.

Emitindo Nota

Analisando Produto

Figura 8.2 – Exemplos de estados
Fonte: Elaboração da autora (2008).

b) Estado inicial – O estado inicial de um objeto ocorre quando ele é criado. Ele é representado por um pequeno círculo fechado. Indica a partir de onde o DTE deve ser lido. Para cada DTE, você só tem **um** estado inicial.

Figura 8.3 – Estado inicial
Fonte: Elaboração da autora (2008).

c) **Estado final** – O estado final indica o final do ciclo de vida de um objeto. Ele é representado por um círculo eclipsado. O estado final é opcional e pode existir mais de um em um mesmo DTE.

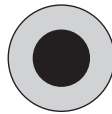



Figura 8.4 – Estado final
Fonte: Elaboração da autora (2008).

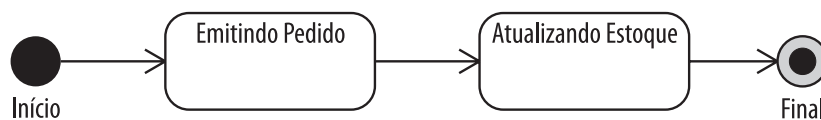



Figura 8.5 – Exemplo DTE
Fonte: Elaboração da autora (2008).

No diagrama da figura 8.5, o estado final acontece logo depois da atualização do estoque que modifica o estado do objeto em estoque.

d) **Transição** – Quando a ação ou atividade de um estado está completa, o fluxo de controle passa ao estado seguinte de ação. Especifica-se esse fluxo utilizando transições para mostrar o caminho de um estado de ação ou de atividade para o estado seguinte.

Graficamente você representa a transição por uma linha simples com uma direção. As transições não ativadas podem ter condições de proteção, significando que a transição será iniciada somente se essa condição for satisfeita.

Imagine quando você está no banheiro pronto para iniciar o banho. Automaticamente você pensa em duas possibilidades: o estado do chuveiro está ligado ou desligado. Em outras palavras: você pode representar isso em um diagrama de estados. Nesse DTE, o evento é girar a torneira; e as ações são abrir e fechar.

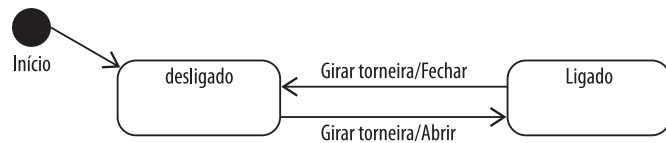



Figura 8.6 – Exemplo DTE tomar banho
Fonte: Elaboração da autora (2008).

Furlan (1998) define quatro possibilidades de eventos em um DTE:

- **Recibo de sinal explícito do outro objeto** – São gatilhos de uma transição (recibo de mensagens). O objeto recebe um sinal de outro objeto e muda de estado. Neste tipo de evento, o objeto que envia a mensagem fica esperando a sua finalização.

- **Passagem de período designado de tempo** – O evento acontecerá após um determinado período de tempo, disparando a mudança de estados. Nesse tipo de evento, utiliza-se a cláusula *after*. Por exemplo: se você tiver o evento *after* (1 minuto), significa que o evento será executado um minuto depois de o objeto entrar no estado atual.

- **Uma condição tornando-se verdadeira** – É mostrada uma condição de guarda em uma transição de estados. Nesse caso, você utiliza a cláusula *when*, por exemplo, *when* (quantidade < 5), e então a transição é disparada se a quantidade for menor que 5.

- **Recibo de chamada de operação pelo próprio ou por outro objeto** – É mostrada como uma assinatura de evento em transição de estado. Um objeto solicita um serviço a outro objeto.

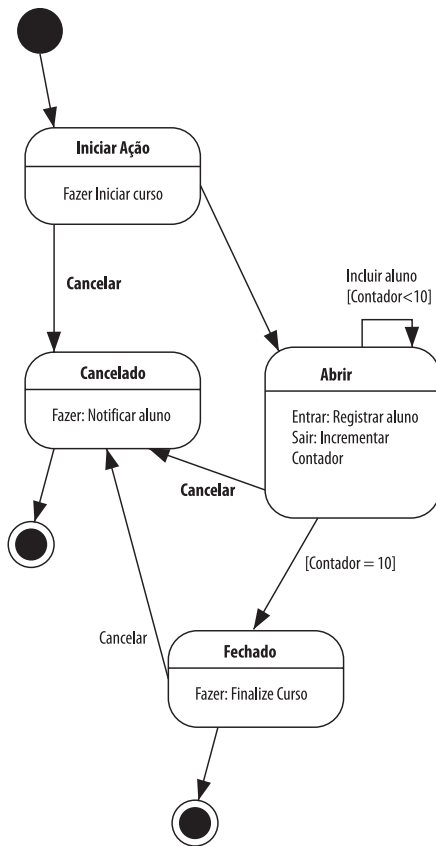



Figura 8.7 – DTE para matrícula de um curso
Fonte: Esmin (1999).

Quando você possui uma ação, ela possui um processo, que é a transição do estado. Essa transição normalmente é um processo de curta duração e que não pode ser interrompido.

Muitas vezes, em vez de ações você pode ter atividades relacionadas. Quando isso acontece, você também possui um processo associado, mas o processo está associado a um estado. O processo é mais duradouro e a atividade pode ser eventualmente interrompida por um evento.

A condição de guarda é uma expressão de valor lógico usada em uma transição. Você pode definir a condição utilizando-se parâmetros, referências e ligações da classe.

Observe na figura 8.7: só será incluído um aluno no curso se o contador for menor que 10.

Quando você define uma condição de guarda, ela só é disparada se o evento associado ocorrer e se a condição for verdadeira.

A condição de guarda sempre aparece no DTE com o uso de colchetes, como:

> Realizar_saque (quantia) [quantia=saldo]/sacar (quantia)

No diagrama da figura 8.8, Lima (2005) apresenta um DTE para Pedidos. O pedido pode ser confirmado ou cancelado pelo cliente. O estado Pendente possui uma ação reflexiva Pedidos Pendentes. Ao finalizar o Pedido, o estado do Pedido estará alterado.

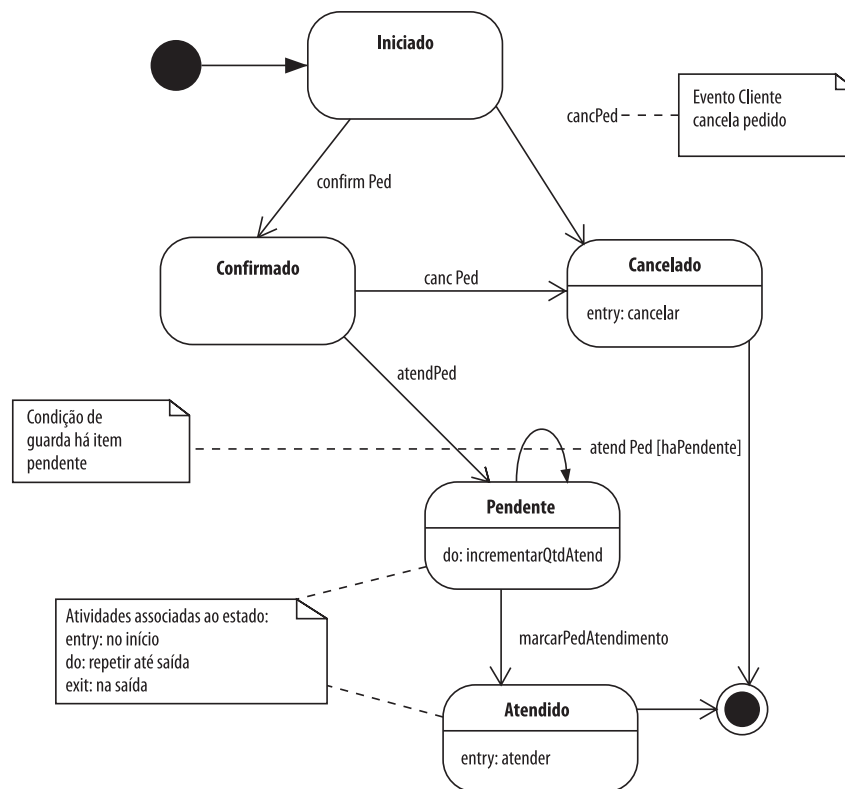



Figura 8.8 – DTE Emitir Pedido
Fonte: Lima (2005).

Lembre-se, utilize o diagrama de transição de estados para:

- descrever o comportamento de um objeto ao longo de vários casos de uso;
- modelar objeto dotado de comportamento muito dinâmico.

Seção 2 – Modelo de atividades

O diagrama de atividades é o quarto diagrama responsável pela descrição dos aspectos dinâmicos de um caso de uso. O diagrama de atividade modela a lógica utilizada em um caso de uso.

Neste caso, o diagrama de atividade permite apresentar interações, decisões e passos executados em paralelo impossíveis de representar somente com o caso de uso.

Pode ser utilizado para descrever um processo de negócio, a partir da necessidade de entender melhor um problema. O diagrama vai permitir que você entenda melhor o comportamento do sistema, no decorrer dos diversos casos de uso.

Esse modelo também é utilizado para especificar a programação com ***multithreading.***

O diagrama de atividade lembra muito o antigo fluxograma, lembra-se dele?

O diagrama de atividade deve ser usado em situações em que todos ou a maioria dos eventos representam a conclusão das ações geradas internamente – ou seja, os fluxos processuais de controle – e em situações onde acontecem eventos assíncronos.

O diagrama permite escolher a ordem pela qual as coisas devem ser feitas, indicando as regras essenciais de sequência que devem ser seguidas.

O conceito de *multithreading* utiliza o conceito de *thread* que é considerado um processo leve, em que o espaço de endereçamento é compartilhado por vários programas. No ambiente *multithreading*, não existe a ideia de programas associados a processos, mas a *threads*. O processo nesse ambiente tem um *thread* de execução, mas compartilha o espaço de endereçamento com inúmeros outros *threads* (MACHADO, 2002).

Para Furlan (1998), os diagramas de atividades podem modelar o sistema de duas maneiras:

- Para modelar o fluxo de trabalho – As atividades são focalizadas conforme visualizadas pelo ator. Por exemplo, no fluxo de trabalho de processamento de um pedido incluirá classes como Pedido e Cobrança. As instâncias dessas duas classes serão produzidas por determinadas atividades (processar pedido criará um objeto Pedido, por exemplo); outras atividades poderão alterar esses objetos (por exemplo, Enviar pedido modificará o estado do objeto Pedido a ser preenchido).

- Para modelar uma operação – Os diagramas são empregados como fluxogramas. O diagrama permite visualizar, especificar, construir e documentar o comportamento de qualquer elemento da modelagem. Os diagramas de atividades podem ser anexados a classes, interfaces, componentes, nós, casos de uso e colaboração.

Na figura 8.9, observe os possíveis componentes de um diagrama de atividades.

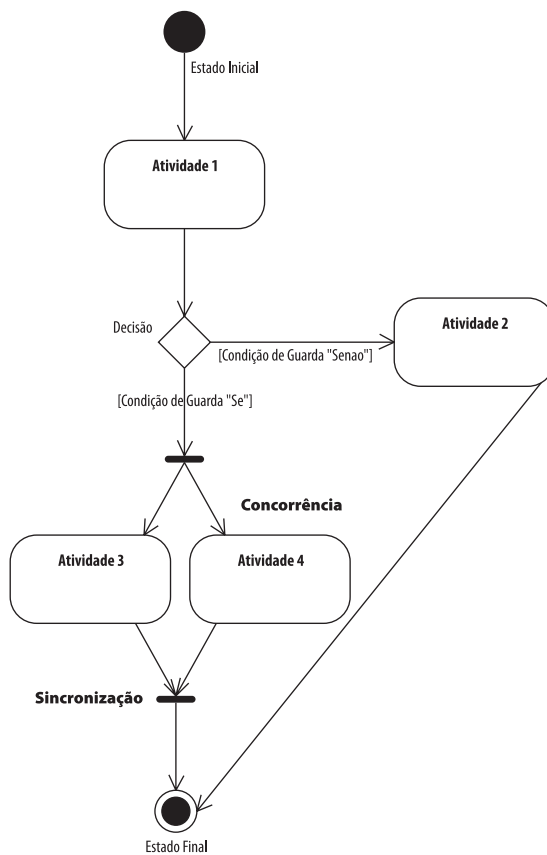



Figura 8.9 – Modelo de um diagrama de atividades
Fonte: Elaboração da autora (2008).

O diagrama sempre tem um estado inicial e um estado final. A transição de término liga um estado a outro, ou seja, o término de um passo é o início de outro. Cada uma das ações é disparada no momento em que ocorre o término do evento anterior.

Os pontos de ramificações são pontos em que, a partir de uma transição de entrada, você pode ter várias transições de saída. É o caso das condições de guarda do diagrama. Os objetos apresentados no diagrama podem ser:

- **Atividade** – Representada pelo retângulo ovalado.
- **Objeto** – Representado por um retângulo.

- **Seta cheia** – Relação de precedência entre atividades.
- **Seta pontilhada** – Consumo ou produção de objeto por atividade.
- **Linha horizontal cheia** – Ponto de sincronização.
- **Pequeno círculo cheio** – Estado inicial.
- **Pequeno círculo eclipsado** – Estado final.

No diagrama, você percebe que existem dois fluxos paralelos (atividade 3 e 4) e que não existe limitação para o número de processos paralelos, pois a sincronização desses fluxos acontece pelo uso de uma barra paralela.

A barra pode ser utilizada para bifurcação ou junção. Se for uma barra de junção (*join*) (no diagrama aparece como sincronização), dois ou mais fluxos de transição serão unidos em um único fluxo. Se for uma barra de bifurcação a partir de uma transição de entrada, são criados dois ou mais fluxos paralelos (no diagrama aparece a bifurcação na condição de guarda "se").

Na figura a seguir, você vê o diagrama de atividade para o caso de uso realizar saque do sistema de caixa eletrônico.

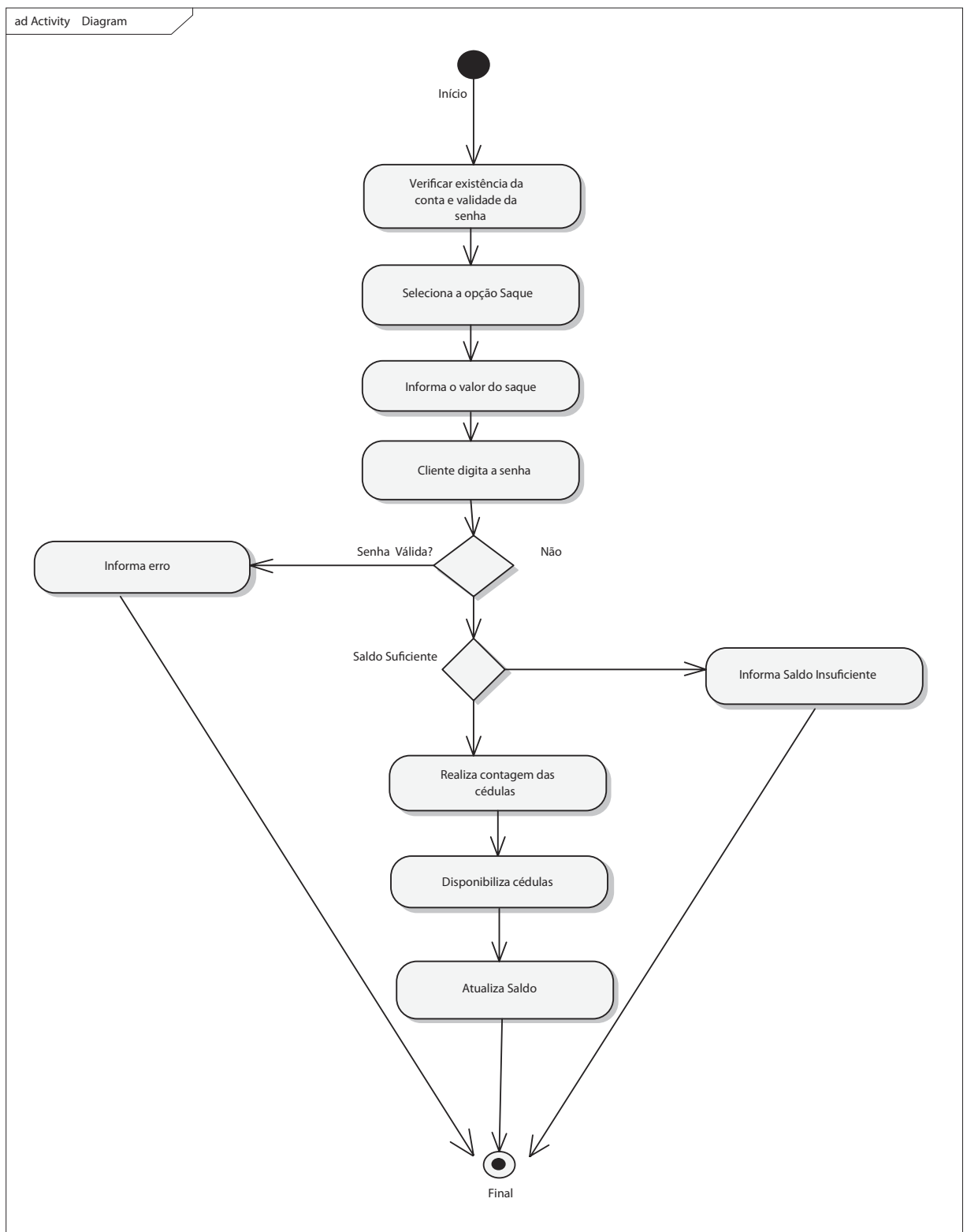



Figura 8.10. – Diagrama Realizar Saque do Sistema de Caixa Eletrônico
Fonte: Elaboração da autora (2008).

É possível observar pelo diagrama a sequência de atividades que devem ser realizadas e a precedência de cada atividade.

Na figura a seguir, é apresentado o diagrama de atividade para o caso de uso Gerar Contrato de Aluguel do sistema Imobiliário:

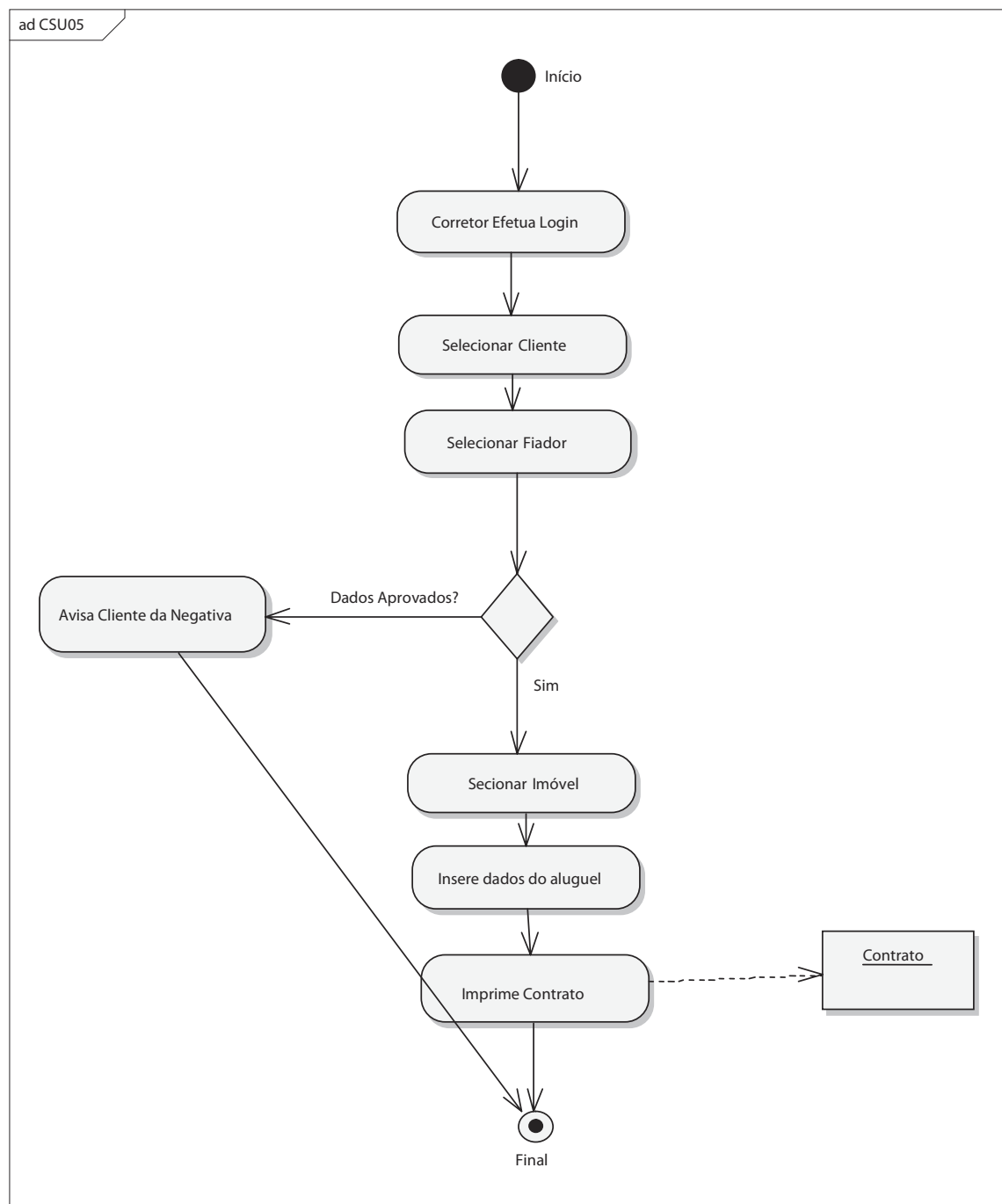



Figura 8.11 - Diagrama Gerar Contrato de Aluguel do Sistema Imobiliário
Fonte: Elaboração da autora (2008).

Lembre-se, utilize o diagrama de atividade:

- Para explicitar comportamentos paralelos;
- na modelagem de *workflow*;
- em caso de programação com *multithreading*;
- para melhor entendimento de *workflow* entre vários casos de uso.

Workflow – descreve tarefas dos processos de negócio (conjunto de uma ou mais atividades relacionadas que, coletivamente, atingem um objetivo) em um nível conceitual necessário para compreender, avaliar e reprojetar o processo de negócio (BORTOLI, 2004).

Seção 3 – Considerações sobre o uso da orientação a objetos

Durante este estudo não foram apresentados todos os diagramas e todas as visões possíveis da UML. Na verdade, foram privilegiadas as visões consideradas fundamentais para qualquer desenvolvimento de *software*.

A complementação pode ser feita por meio da leitura do livro UML: guia do usuário escrito por Booch, Rumbaugh e Jacobson, em 2000.

Ao finalizar esta unidade, é importante retomar um tema importante: quais são os benefícios do uso da orientação a objetos? Pode-se afirmar que o uso desse paradigma proporciona o melhor gerenciamento das funções do sistema, o aumento da produtividade pelo uso da reutilização, a melhor qualidade dos serviços oferecidos e a facilidade do mapeamento do mundo real versus o mundo computacional.

Apesar de todas essas vantagens, o mercado ainda não absorveu completamente essa metodologia. Por quê?

Furlan (1998) lista alguns motivos para tal quadro:

- incerteza;
- falta de mão de obra qualificada;

- ferramentas imaturas;
- O investimento da empresa já realizada em ferramentas não orientadas a objetos.

A introdução da UML, suas facilidades e o grande número de ferramentas que suportam sua notação vêm gradativamente revertendo esse quadro. O grande potencial de comunicação e reaproveitamento pela reutilização de modelos em futuras aplicações tem aproximado empresas de desenvolvimento do modelo orientado a objetos.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas.

Síntese

Com esta unidade, você complementou seu conhecimento sobre a representação dinâmica do sistema por meio de seus diagramas de atividade e estado.

Os fluxos de estado são excelentes na descrição de fluxos complexos, permitindo a visualização da execução do caso de uso.

O diagrama de estado descreve muito bem o comportamento de um único objeto, mas quando o comportamento envolve diversos objetos é mais adequado o uso do diagrama de atividades.

O diagrama de atividades suporta a descrição de comportamentos paralelos modelando o fluxo de trabalho, principalmente quando diferentes casos de uso interagem entre si ou utilizam multiprocessamento. Outra situação de uso comum para o diagrama de atividade é seu uso para entender o comportamento, dependências das ações e as próprias ações necessárias na realização do caso de uso.

Atividades de autoavaliação

Leia com atenção os enunciados e realize as questões propostas.

1) Complete as afirmações a seguir.

a) Esta modificação é chamada de ________________ entre estados.

b) Os _______________ representa o resultado de atividades executadas por um objeto.

c) O _______________ indica o final do ciclo de vida de um objeto.

d) Os _______________ de um sistema modificam seu estado de forma análoga a objetos do mundo real.

2) Indique se os conceitos a seguir fazem parte do diagrama de estados (DTE) ou do diagrama de atividade (DA).

a) (   ) O diagrama mostra os estados de um objeto.

b) (   ) O diagrama explicita comportamentos paralelos.

c) (   ) O diagrama mostra os eventos ou as mensagens que causam uma transição.

d) (   ) O diagrama apoia o entendimento de *workflow* entre vários casos de uso.

3) Relacione os conceitos a seguir. Observe que uma mesma opção pode se repetir.

A) *After*

B) Estado inicial

C) *When*

D) *Transient*

E) *Join*

a) (   ) É utilizado em um DA para indicar uma bifurcação.

b) (   ) Ocorre quando o objeto é criado.

c) (   ) Utilizado para designar a paassagem de um período predeterminado de tempo.

d) (   ) É utilizado para representar uma condição de guarda.

e) (   ) É utilizado em um DA para indicar uma sincronização.

4) Construa o diagrama de atividades do caso de uso Agendar Horário da Clínica Bem-Estar a partir da visão do ator Atendente.

Empresa : Clínica Bem-Estar

1) Função: fomos contratados para analisar seu processo atual e verificar como expandir suas operações e melhorar seu nível de serviço.

Histórico:

A clínica, fundada há 5 anos, atua no atendimento clínico pediátrico.

A clínica possui 34 médicos cadastrados em diferentes especialidades como: cardiologia, clínica geral, dermatologia etc. Todos os médicos utilizam internet e e-mail. A faixa etária predominante é de 30, 35, 40, 42, 44 e 48 anos. Todos os médicos são aptos do ponto de vista físico.

O paciente pode ser atendido de forma particular ou por convênios. Os convênios atendidos são o Bruxtr, Vpfzm e UIOlk.

Cada médico faz 3 plantões semanais de 4 horas seguidas; as consultas possuem um intervalo de 30 minutos. Existe a possibilidade de a consulta ser de retorno, nesse caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde são marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio. Trabalham há 3 anos na clínica com planilhas Excel.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente, que contém nome, endereço, telefone, data de nascimento, convênio.

O médico, ao atender o paciente, preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

A clínica possui de 700 a 800 fichas, sendo que cerca de 600 são de atendimento por convênio.

O gerente da clínica está ansioso, pois não consegue controlar questões relacionadas ao número de pacientes atendidos por convênio e particular, médicos mais procurados e picos de movimento.

Volume de atendimentos: 56 por dia.

Outra questão de interesse é manter um controle de laboratórios conveniados, pois o médico poderia indicar o laboratório já no momento da prescrição.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar em:

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. São Paulo: Campus, 2002. (Ler capítulos 10 e 11.)

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML:** guia do usuário. São Paulo: Campus, 2000. (Ler capítulos 18 e 19.)

BORTOLI, A. **O uso do *workflow* para apoiar a elicitação de requisitos**. *Workshop* em engenharia de requisitos. 2004

FURLAN, José. **Modelagem de objetos**. São Paulo: Makron Books, 1998. (Ler capítulo 6.)

UNIDADE 9

9

RUP e ICONIX

Objetivos de aprendizagem

- Entender o que é o RUP, seus elementos e conceitos.
- Discutir e analisar nuanças do processo de desenvolvimento orientado a objetos utilizando-se o RUP.
- Entender como o RUP colabora para a estruturação efetiva de tarefas e fluxos de trabalho de profissionais, atuando em equipes de desenvolvimento de software.
- Compreender o processo ICONIX e seus componentes.

Seções de estudo

Para início de estudo

Quando você lê um artigo sobre empresas de desenvolvimento de *software,* é raro encontrar discussões sobre a dificuldade da empresa em dominar uma tecnologia como um banco de dados, uma linguagem de programação ou mesmo um sistema operacional.

Por outro lado, você vai encontrar dezenas de matérias relatando os problemas relacionados a cronogramas, entregas, produtividade da equipe, qualidade do produto, uso de metodologias e padronização das diferentes etapas.

Ensinar UML a uma equipe de projeto pode não ser uma tarefa tão árdua, como propor a essa equipe todo um processo de desenvolvimento voltado para uma metodologia, utilizando essa notação.

O *Rational Unified Process* (RUP) é um processo de engenharia de *software* que pretende aumentar a produtividade da equipe, oferecendo práticas eficientes executadas por meio de diretrizes, *templates* e orientações sobre ferramentas para todas as atividades críticas de desenvolvimento de *software*s.

O ICONIX, no entanto, é um método menos complexo, com um volume menor de documentação que o RUP, mas que apresenta grande aceitação no mercado.

Seção 1 – Aonde se quer chegar?

Quando olhamos ao redor, percebemos nossa total dependência dos sistemas de informação. Mas o que é feito em nossa vida moderna que não envolva recursos computacionais?

Poucas são as atividades que nos restam em que não temos como apoio um *software*. Além da dependência, temos os riscos

envolvidos nesse problema complexo. Antigamente se imaginava arriscado um *software* existente em uma aeronave que, ao falhar, poderia fazer com que ela caísse. Ou um sistema de controle de uma usina nuclear.

Hoje os riscos extrapolam questões vitais. Imagine os riscos financeiros provenientes de milhares de transações bancárias, os riscos em uma fraude eleitoral ou mesmo os riscos existentes em um *software* de defesa de um país, como a Inglaterra.

O desenvolvimento de *software* é uma atividade de alto risco. Entre lucros gerados nessa atividade, muitos foram os prejuízos, na maioria das vezes causados pelo mau planejamento, pela má gerência e pela baixa qualidade do produto final.

Mas como gerenciar o processo de desenvolvimento de *software* aumentando sua qualidade se a empresa de desenvolvimento de *software* não conhece seu próprio processo de desenvolvimento?

Segundo Booch (2000), um **processo** é um conjunto de passos parcialmente ordenados com a intenção de atingir metas. A meta neste caso é a entrega eficiente e previsível de um produto de *software* que atenda, de forma completa, todas as necessidades do usuário.

O processo define: quem está fazendo o quê, quando e como está sendo feito e as ações para atingir uma determinada meta.

Mas como inserir qualidade no processo? A qualidade passa pelo uso de metodologias e reconhecimento formal do processo.

Apenas modelar o sistema não é o suficiente. O uso de uma notação voltada à orientação a objetos como a *Unified Modeling Language* (UML) pode colocar sua empresa na vanguarda em termos de notação, porém pouco garante quanto à qualidade de todo o processo.

A UML é uma linguagem de especificação. O seu uso garante a confecção de diagramas precisos. Mas, se esses diagramas não forem usados de forma sistemática, documentados e servirem como pontos de controle e avaliação do processo, de pouco servirão para a adequação de qualidade do produto.

Em resumo, somente descrever os diagramas não é o suficiente para garantir um processo de desenvolvimento com qualidade. É necessário o uso de uma metodologia que unifique o esforço da equipe dentro de um processo formal e mensurável.

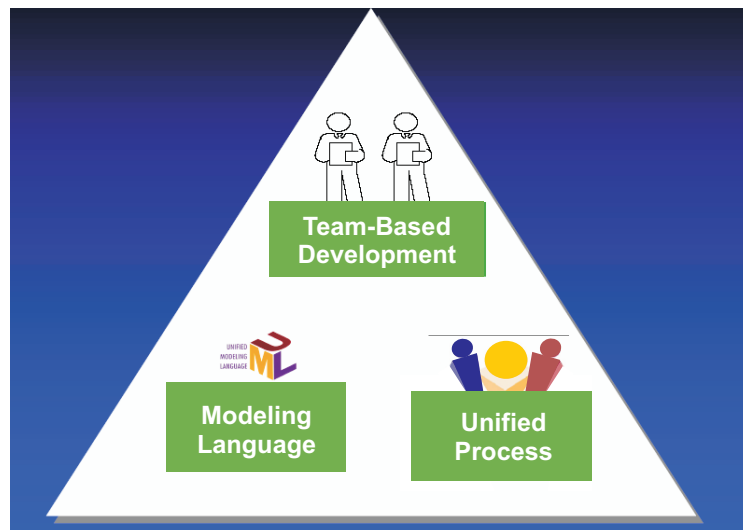



Figura 9.1 – Métodos
Fonte: IBM (2005).

Como está estruturado o *Rational Unified Process (RUP)*?

O RUP usa a abordagem da orientação a objetos em sua concepção. Sua projeção e documentação utilizam a notação UML para especificar, modelar e documentar artefatos com os quais se procura ilustrar os processos em ação.

O RUP foi desenvolvido pela *Rational Software Corporation*, sendo então um método proprietário de desenvolvimento de *software*.

Segundo IBM (2005), o RUP se apresenta como um processo de engenharia de *software* que oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento, permitindo o gerenciamento eficiente do processo.

Sua meta é garantir a produção de *software* de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e orçamento previsíveis (IBM, 2005).

Kroll e Kruchten (2003) apresentam três definições para o RUP:

- É uma maneira de desenvolvimento de *software* que é iterativa, centrada à arquitetura e guiada por casos de uso.

- É um processo de engenharia de *software* bem definido e bem estruturado. O RUP define claramente quem é responsável pelo que, como as coisas devem ser feitas e quando fazê-las. Além disso, também provê uma estrutura bem definida para o ciclo de vida de um projeto RUP, articulando claramente os marcos essenciais e pontos de decisão.

- É também um produto de processo que oferece uma estrutura de processo customizável para a engenharia de *software*. O produto IBM RUP suporta a customização e autoria de processos e uma vasta variedade de processos ou configuração de processos. Essas configurações do RUP podem ser criadas para suportar equipes grandes e pequenas e técnicas de desenvolvimento disciplinadas ou menos formais.

Quais são as diretrizes do RUP?

O RUP apresenta características próprias no processo de desenvolvimento (IBM, 2005):

- **O uso de um processo iterativo** – O problema e a solução são organizados em pequenas partes e para cada pequena parte do sistema é feita uma iteração. A iteração segue o modelo sequencial tradicional, com identificação de necessidades, análise, projeto, implementação e desenvolvimento.

- **O processo é incremental** – Cada interação acrescenta incrementalmente novas características ao produto. Assim, a cada ciclo a solução delineada é aperfeiçoada.

- **Dirigido por casos de uso** – O processo é guiado pelas interações entre o sistema e os usuários. Todos os casos de uso de um sistema compõem a especificação funcional do sistema (modelo de casos de uso), ou seja, definem os requisitos e objetivos do sistema do cliente. Assim os casos de uso associam todos os *workflows* de forma conjunta, dirigindo várias atividades de desenvolvimento como a criação e validação da arquitetura do sistema, a criação de casos de teste, o planejamento das iterações, a criação de documentação do usuário e a implantação do sistema. Os casos de uso sincronizam conteúdo dos modelos criados em cada *workflow*.

- **Centrado na arquitetura** – Neste caso, o processo focaliza o desenvolvimento inicial e a linha de base da arquitetura de *software* utilizando-se de relacionamentos claros entre componentes da arquitetura. Você pode ver o sistema como a soma de diversas partes menores (componentes). Cada componente possui a funcionalidade necessária. Sob o ponto de vista da aderência ao negócio ou em termos de implementação, esses componentes se comunicam por meio de interfaces. O uso de componentes torna a manutenção e a reutilização muito mais eficientes.

O **RUP** foi desenvolvido para ser aplicável a qualquer tipo de projeto. Na literatura, é assumido como um *framework* genérico para processos de desenvolvimento.

Você pode encontrar esse padrão no site do *Rational Software Corporation*. Para que você o utilize da forma mais adequada, ele dever ser configurado de acordo com as necessidades da empresa ou mesmo do projeto.

Seção 2 – Quais são as fases do RUP?

As fases de um projeto podem ser definidas como o período de tempo necessário entre dois marcos de progresso de um processo, em que um objetivo é alcançado, artefatos são concluídos e decisões sobre a etapa seguinte são tomadas.

Segundo o site da IBM Rational Software, o RUP é composto por quatro fases:

a) **Concepção ou inicial** – Estabelece o caso de negócio para o projeto. É estabelecido o escopo e a viabilidade econômica do projeto.

b) **Elaboração** – Nesta fase são estabelecidos o plano de projeto e uma arquitetura sólida. Os principais riscos são eliminados. Ao final da etapa, é estabelecida a arquitetura, a partir da qual o sistema vai evoluir.

c) **Construção** – Nesta etapa o sistema é desenvolvido. O produto completo é desenvolvido iterativamente.

Para saber

Workflow – É a automação de processos de negócio, em que as atividades são passadas de um participante para o outro, de acordo com um conjunto de regras definidas.

Framework – No desenvolvimento do *software*, um *framework* é uma estrutura de suporte definida em que um outro projeto do *software* pode ser organizado e desenvolvido. Tipicamente, um *framework* pode incluir programas de apoio, bibliotecas de código, linguagens de *script* e outros *software*s para ajudar a desenvolver e juntar diferentes componentes do seu projeto.

d) Transição – Fornece o sistema aos seus usuários finais, normalmente por uma versão beta do sistema. Se necessário, no final da fase pode ser iniciado um novo ciclo de desenvolvimento para a evolução do produto.

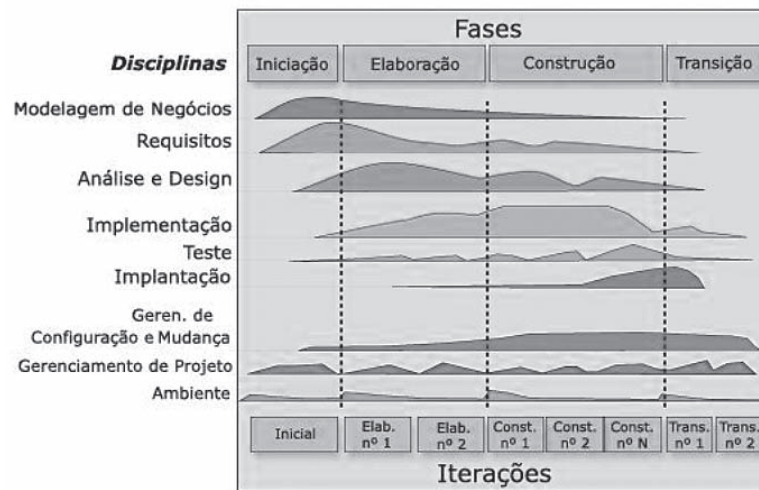



Figura 9.2 – Fases do RUP
Fonte: Adaptado de IBM (2005).

A cada fase ocorre um ciclo completo de desenvolvimento, da análise ao produto executável. Cada fase é finalizada com um marco de progresso (*milestone*), no qual são verificados se os objetivos da fase foram alcançados.

A cada iteração (elaboração, construção, transição) são realizadas atividades listadas à esquerda do gráfico. As curvas apresentadas no gráfico indicam o esforço dedicado às atividades a cada fase do projeto.

As interações são ciclos completos de desenvolvimento. Cada interação passa por vários fluxos de trabalho do processo com ênfases diferentes.

Como foi comentado na primeira seção, o RUP é direcionado pelos casos de uso. Mas como isso acontece?

A cada iteração são identificados e especificados os casos de uso mais relevantes. São feitos então a análise e o projeto dos casos de uso, sempre usando a arquitetura como guia. A partir desse ponto são implementados os componentes que realizam o que foi projetado e, finalmente, verifica-se se os componentes que satisfazem o que foi previsto como objetivos de cada caso de uso.

Os casos de uso são usados durante todo o processo:

- O caso de uso é utilizado para especificar os requisitos.
- Na etapa de análise, projeto e implementação, os casos de uso são executados.
- Na etapa de testes, você vai verificar se o sistema realiza o que está descrito no modelo de casos de uso.
- E, finalmente, os casos de uso são a ferramenta fundamental no planejamento e acompanhamento de todas as iterações.

Seção 3 – Quais são os elementos do RUP?

Segundo Booch (2000), quando você utiliza a metodologia RUP, percebe-se imediatamente a existência de cinco elementos principais: os papéis, os artefatos, as atividades, a disciplina e os fluxos de atividade (subdividido em fluxos principais e fluxos de apoio).

a) Papéis – Um papel (ou perfil) define o comportamento e as responsabilidades de um determinado indivíduo ou grupo de indivíduos que trabalham dentro de uma equipe.

É importante que você perceba que os papéis não são indivíduos, mas sim o papel que ele assume no processo. Veja alguns papéis:

O analista de sistema em que o indivíduo que assume esse papel coordena a obtenção dos requisitos, a modelagem dos casos de uso e a definição do escopo do projeto.

O projetista de testes, responsável por toda a estruturação da etapa de testes, do planejamento à avaliação dos resultados dos testes.

Assim, em um projeto você pode ter vários indivíduos executando um mesmo papel.

b) Atividade – Uma atividade é uma unidade de trabalho que um indivíduo executa quando está exercendo um determinado papel, produzindo um resultado para o contexto do projeto. A atividade é composta de objetivos, passos, entradas e saídas, o responsável pela atividade e os guias e padrões que devem ser seguidos pela atividade.

Exemplo: uma atividade pode ser a definição dos atores, a definição dos casos de uso ou mesmo conduzir uma etapa de testes.

c) Artefato – O artefato é o produto de um projeto. É o resultado de uma etapa. Apesar de aparecerem como resultado do desenvolvimento do projeto, podem ser utilizados como entrada de uma atividade e ainda assim no final da atividade vai gerar outro artefato como saída. Normalmente o artefato é baseado em modelos de como devem ser feitos ou mesmo por documentos que possuem já um formato padronizado.

Exemplo: um artefato pode ser um modelo de caso de uso, um caso de uso, um código-fonte etc.

d) Disciplinas – Uma disciplina mostra todas as atividades que você deve realizar para produzir um determinado conjunto de artefatos. As disciplinas são descritas em nível geral, com um resumo de todos os papéis, atividades e artefatos envolvidos.

Em alguns pontos do projeto, você precisa de um detalhamento maior. Neste caso, ocorre o detalhamento da disciplina por meio do fluxo de trabalho.

Um fluxo de trabalho é uma sequência de atividades que são executadas para a produção de um resultado para o projeto. Fluxos de trabalho podem ser representados por diagramas de sequência, diagramas de colaboração e diagramas de atividades da linguagem UML.

e) Fluxos de trabalhos – O RUP é desenvolvido baseando-se em nove fluxos de trabalho de processo, que podem ser divididos em fluxos principais e fluxos de apoio, definidos em Booch (2000). Acompanhe, a seguir.

Fluxos principais

- **Modelagem do Negócio (*Business Modeling*)** – Envolve o entendimento da estrutura e dinâmica da organização cliente, garantindo que clientes, usuários e desenvolvedores tenham a mesma visão da organização para a qual será feito o desenvolvimento.

- **Requisitos (*Requirements*)** – Esta disciplina visa estabelecer e manter concordância com os clientes e outros envolvidos, sobre o que o sistema deve fazer, utilizando-se para isso dos casos de uso. Também é definida a delimitação do sistema, são definidas as bases para planejamento do conteúdo técnico das interações, estimativas de custo, o tempo de desenvolvimento do sistema e a definição da interface do usuário com o sistema, focando nas necessidades e metas dos usuários.

- **Análise e Projeto (*Analysis and Design*)** – Envolve a tradução dos requisitos numa especificação que descreve como implementar o sistema, adaptando o *design* para que corresponda ao ambiente de implementação, projetando-o para fins de desempenho. É nesse momento que são descritas as diferentes visões do sistema.

- **Implementação (***Implementation***)** – Envolve o desenvolvimento de código: classes, objetos etc., teste de unidades e integração de subsistemas.

- **Teste (***Test***)** – Descreve todos os casos de teste, procedimentos e medidas para o acompanhamento dos erros ocorridos.

- **Entrega (***Deployment***)** – Abrange a configuração do sistema a ser entregue (empacotamento, distribuição, instalação, treinamento de usuários, planejamento e condução de beta teste). São descritos três modos de implantação de produto: a instalação personalizada, o produto em uma forma "compacta" e o acesso ao *software* por meio da internet.

Fluxos de atividades de apoio

- **Gerência de Projeto** (*Project Management*) – Enfatiza principalmente o gerenciamento de risco, o planejamento de um projeto iterativo por meio do ciclo de vida e de uma iteração particular. O monitoramento do progresso de um projeto iterativo e o uso de métricas. Aspectos relacionados à gerência de pessoal, aos contratos e ao orçamento não são observados.

- **Gerência de Configuração e Mudanças** (*Configuration and Change Management*) – O fluxo controla as modificações mantendo a integridade dos artefatos do projeto. Envolve a identificação dos itens de configuração, a restrição de mudanças, a auditoria das mudanças feitas e a definição e o gerenciamento das configurações desses itens.

- **Ambiente** (*Environment*) – Envolve a infraestrutura necessária para que o desenvolvimento ocorra. A meta é oferecer à organização o ambiente de desenvolvimento de *software,* processos e ferramentas que darão suporte à equipe de desenvolvimento.

Quais são os modelos do RUP?

Quais os prós e contras do *Rational Unified Process?*

A discussão sobre vantagens e desvantagens do RUP é bastante acirrada.

Uma grande vantagem a ser considerada é o uso de princípios de engenharia de *software* na sua abordagem de desenvolvimento iterativa, incremental, orientada a requisitos e baseada em arquitetura. O sistema desenvolvido com o RUP tende a tolerar melhor mudanças de requisitos e alterações no produto.

Os componentes desenvolvidos ao longo de um projeto podem ser reutilizados em outros projetos, diminuindo o tempo de desenvolvimento e, consequentemente, os custos para o cliente.

Um dos aspectos mais importantes é a capacidade de gerência por meio de fases e *millestones* incorporados ao projeto.

Ao lado de suas vantagens, também temos limitações do método, entre elas o fato de o RUP não abordar a gestão de pessoal e contratos, como a ferramenta ser um sistema de hipertexto dificultando as integrações com outras ferramentas.

Talvez um dos fatores mais críticos do método seja o fato de o método ser previsto para qualquer porte de empresa. A implantação em empresas de pequeno porte tem se mostrado, no entanto, difícil pelo volume de responsabilidades e a quantidade de atividades de cada fluxo de atividade.

Quer conhecer mais?

Para aprofundar seus conhecimentos sobre esta unidade, acesse a Midiateca no EVA. Leia o texto **Um método de desenvolvimento de sistemas de grande porte baseado no processo RUP**. Com ele, você vai poder avaliar melhor o processo RUP.

Seção 4 – ICONIX

O ICONIX é uma metodologia de desenvolvimento de software com características interativas e incrementais. Classificar o ICONIX é difícil, pois por um lado possui uma veia tradicional com um processo bem definido, por outro lado aproxima-se dos métodos ágeis procurando a redução da documentação e a simplicidade no processo.

Mastelari (2004) define ICONIX Software Engineering como um processo enxuto e robusto voltado para o trabalho com equipes pequenas e desenvolvimentos de tamanho pequeno e médio.

O processo ICONIX teve seu início muitos anos antes da concepção da UML e do processo unificado. Foi elaborado por Doug Rosenberg e Kendall Scott Poe, voltando-se a uma abordagem de orientação a objetos. Silva e Videira (2001) apresentam o ICONIX como uma metodologia prática, intermediária entre a complexidade do RUP e a simplicidade do XP (Extreme Programming). A UML é usada integralmente no método suportando e respondendo a questões impostas pela metodologia por meio de seus diagramas.

São, segundo Borillo (2000), três as características fundamentais no ICONIX:

- O modelo é **iterativo e incremental** e, portanto, várias iterações acontecem da definição do modelo de domínio à identificação dos casos de uso.

- **Rastreabilidade** – Todos os passos do processo referenciam os requisitos. O modelo permite então verificar em todas as fases se os requisitos foram atendidos. Desta forma, pode-se determinar qual o impacto que a alteração de um requisito tem em todos os artefatos do sistema.

- **Aerodinâmica da UML** – A metodologia incorpora o uso da UML por meio de diagramas. Os mais usados são: os diagramas de casos de uso, diagramas de sequência e colaboração, diagramas de robustez e diagramas de classes.

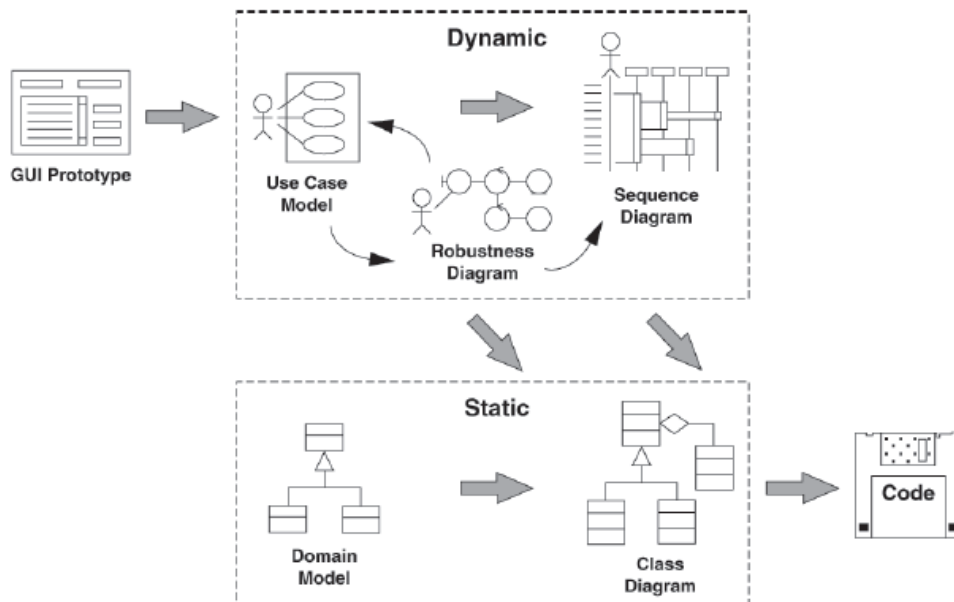



Figura 9.3 – Processo Iconix
Fonte: Rosenberg et al. (2005).

O ICONIX fundamenta-se em dois modelos: o estático e o dinâmico. O modelo estático modela o funcionamento do sistema sem se preocupar com interações e aspectos dinâmicos do sistema. São dois os diagramas usados nessa representação: o diagrama de domínio e o diagrama de classe. O modelo dinâmico apresenta as interações do sistema mostrando a interação do usuário com o sistema. O modelo dinâmico é representado por diagramas de sequência, de casos de uso e o diagrama de robustez.

O ICONIX determina um ciclo com quatro etapas bem determinadas, a análise de requisitos, a análise e o projeto preliminar, o projeto e a implementação.

Análise de requisitos

Na análise de requisitos, ocorre a identificação das necessidades do cliente por meio dos requisitos funcionais. Nessa fase o contato com o cliente é estreito. Segundo Silva e Videira (2001), a tarefa de análise de requisitos consiste em realizar as seguintes tarefas:

- Identificar os objetos do domínio do problema. Nessa identificação utiliza-se o diagrama de classes.
- Desenvolver protótipos da interface, do diálogo e da navegação proporcionando para o cliente o melhor entendimento sobre a proposta.
- Identificar os casos de uso e os atores associados a esses casos de uso. Essa tarefa será realizada por meio dos diagramas de casos de uso.
- Finalizada a identificação dos casos de uso, eles devem ser organizados por meio de grupos que permitam sua melhor compreensão e visualização. Essa estruturação ocorre por meio do diagrama de pacotes.
- Os requisitos funcionais devem ser claramente associados aos casos de uso e aos objetos do domínio, de forma a tornar visível qual caso e classe solucionará o requisito funcional.

Análise e projeto preliminar

Durante a etapa de análise e projeto, são descritos os casos de uso identificando-se seus cenários. Nessa etapa do processo, são construídos os diagramas de robustez e são refinados e finalizados os diagramas de classe.

Projeto

A etapa de projeto permite à equipe a especificação do comportamento esperado nos casos de uso, a identificação dos objetos e atores e as mensagens trocadas entre os elementos. Para essa tarefa, o diagrama de sequência torna-se essencial.

Nesse momento já é possível inserir no diagrama de classes seus atributos e métodos. São assim concluídos os diagramas do modelo estático validando se todos os requisitos prescritos foram atendidos.

Implementação

Para a etapa de implementação, a equipe apresenta seu maior esforço na geração do código, na realização de testes de unidade, integração e aceitação do cliente.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas.

Síntese

Nesta unidade, você teve contato com o RUP e o ICONIX, seus conceitos e principais modelos.

Foi possível perceber a importância do processo incremental iterativo oferecido pelo modelo e a preocupação de tornar a gerência e a manutenção do produto mais eficiente em todas as etapas do desenvolvimento. Você percebeu que dentro do método é fundamental a definição clara dos papéis e das responsabilidades. Um dos pontos fortes do modelo é a padronização e a definição de artefatos para cada etapa do projeto.

A utilização de uma arquitetura baseada nos casos de uso aproxima o usuário final do desenvolvimento, melhorando a qualidade do processo, refinando em etapas bastante iniciais o processo de validação do sistema.

Apesar das controvérsias relacionadas ao seu uso, o RUP tem-se mostrado uma metodologia eficiente no mercado, sendo que sua utilização vem crescendo gradativamente. Um dos fatores mais fortes para isso é a possibilidade de adaptação do modelo às necessidades e exigências da empresa.

Um ponto forte do ICONIX é a identificação e representação do modelo de domínio e dos casos de uso. A partir desse ponto o processo de desenvolvimento passa a ser iterativo e incremental. Os casos de uso são documentados e a ele é anexado o respectivo diagrama de robustez. Na sequência identificam-se novos objetos e detalhes que são incorporados ao diagrama de domínio. Na etapa seguinte, os diagramas de sequência são feitos procurando a identificação de objetos. Na última etapa, operações são adicionadas assim como os atributos ao diagrama de classe.

O ICONIX é uma sugestão de processo de desenvolvimento de software e tem tido uma grande aceitação no mercado. Parte desse sucesso se deve à forma sucinta com que trata a documentação. Outro aspecto relevante é a importância do modelo na etapa de entendimento dos requisitos do cliente. Essa preocupação tem transformado o modelo em um modelo de sucesso.

Atividades de autoavaliação

Leia com atenção os enunciados e realize as questões.

1) Assinale as afirmativas corretas (mais de uma, caso necessário):

a) ( ) O RUP utiliza-se do modelo iterativo para o desenvolvimento do *software*. Isso significa a definição clara de etapas em um ciclo rígido e formal.

b) ( ) O RUP é visto como um produto de processo de engenharia customizável.

c) ( ) O RUP é centrado na construção do produto. Baseia-se fundamentalmente no uso de uma linguagem orientada a objetos.

d) ( ) O RUP é centrado na arquitetura, estabelecendo de forma clara e segura todos os relacionamentos existentes entre seus componentes.

2) Entre os elementos do RUP temos os papéis. Qual a sua importância dentro do modelo? Faça uma pesquisa e descreva dois papéis existentes no modelo.

3) Relacione os fluxos de atividades do RUP.

A. Modelo de negócios

B. Requisitos

C. Análise e projeto

D. Gerência de projeto

E. Gerência de configuração e mudanças

a) ( ) Procura manter a integridade dos artefatos do projeto.

b) ( ) Procura manter uma visão uniforme sobre o projeto para todos os envolvidos.

c) ( ) Descreve as diferentes visões do sistema.

d) ( ) Enfatiza o gerenciamento de riscos.

e) ( ) Neste fluxo é definida a delimitação do sistema.

4) Dos nove modelos oferecidos pelo RUP, defina três que você considera fundamentais para que o projeto seja bem aceito pelo cliente final.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, acesse a Midiateca.

Uma outra metodologia bastante aplicada em empresas de *software* atualmente é o ICONIX ((link ICONIX_guj.pdf). No artigo escrito por José Anízio Maia, você vai perceber que é uma metodologia simples e menos burocrática do que o RUP. Vale a pena conferir!

Para concluir o estudo

Com este livro, você aprendeu os conceitos e particularidades da análise estruturada e da análise essencial.

Conheceu as análises, características e particularidades de implementação que tornam o projeto mais claro, facilitando sua compreensão, inclusive para o usuário final.

O uso das metodologias torna o trabalho do gerente mais objetivo e sua cobrança perante a equipe mais eficiente. Isso se deve pela possibilidade de geração de diferentes tipos de artefatos que podem ser utilizados como marcos de projeto para acompanhamento da equipe.

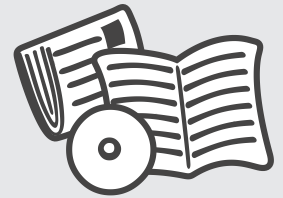
A análise estruturada foi pioneira em termos de aceitação como metodologia de documentação de projetos. Seu uso ainda hoje é expressivo pela facilidade de utilização de sua notação e benefícios advindos de seu uso.

O segundo livro da disciplina dará ênfase à metodologia orientada a objetos fazendo uso da linguagem de notação Linguagem de Modelagem Unificada (LMU). Além de explorar seus diagramas mais utilizados, também será feita uma breve abordagem sobre o *Rational Unified Process* (RUP).

Durante todo o seu estudo nesta disciplina, você foi apresentado a três paradigmas diferentes: a análise estruturada, a análise essencial e a análise orientada a objetos. Você conheceu características e particularidades de implementação de cada um. Foi possível perceber que a orientação a objetos encaixa-se muito bem no paradigma atual de implantação, mas que todos esses tipos de análise, sem exceção, colaboram para que o projeto fique claro para toda a equipe de projeto.

Esperamos que o estudo da disciplina tenha lhe proporcionado a oportunidade de reconhecer a metodologia mais adequada para seus projetos, assim como o entendimento sobre seus conceitos e diagramas.

Professora Vera Schuhmacher

Referências

AMBLER, S. W. **Análise e projeto orientados a objetos**. Rio de Janeiro: Infobook, 1998.

ARAGÃO, S. **Modelagem visual de objetos UML**. São Paulo: [s.e.], 2005.

BECK, K. **Programação extrema explicada**. Porto Alegre: Bookman, 1999.

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. São Paulo: Campus, 2002.

BONA, Cristina. **Avaliação de processos de software:** um estudo de caso em XP e Iconix. 2002. Dissertação (Mestrado em Engenharia de Produção) - Universidade Federal de Santa Catarina. Florianópolis, 2002.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML**: guia do usuário. São Paulo: Campus, 2000.

BORTOLI, A. **O uso do workflow para apoiar a elicitação de requisitos**. Workshop em Engenharia de Requisitos. 2004.

CHEN, Peter. **Gerenciando banco de dados**: a abordagem entidade-relacionamento para projeto lógico. São Paulo: McGraw-Hill, 1990.

COAD, Peter; YOURDON, Edward. **Análise baseada em objetos**. Rio de Janeiro: Campus, 1992.

COMUNIDADE IBM RUP (2005). Disponível em: <http://www-130.ibm.com/developerworks /rational/community>. Acesso: 10 ago. 2005.

COUTO, A. B. **CMMI**: integração dos modelos de capacitação e maturidade de sistemas. Rio de Janeiro: Ciência Moderna, 2007.

DEMARCO, Tom. **Análise estruturada e especificação de sistema**. Rio de Janeiro: Campus, 1989.

ENGHOLM JUNIOR, Helio. **Engenharia de software na prática**. São Paulo: Novatec, 2010.

ESMIN, A. A. A. Modelando com UML - Unified Modeling Language. **II SECICOM** - Universidade Federal de Lavras – UFLA. Lavras, nov., 1999.

FIGUEIRA, J. M.; COSTA, W. S. **A importância de utilizar UML para modelar sistemas**: estudo de caso. Disponível em: <http://www.cefetsp.br/edu/sinergia/6p10c.html>. Acesso em: 10 ago. 2005.

FOWLER, M. **UML Distilled Second Edition:** a brief guide to the standard object modeling language. Nova Jersey: Prentice-Hall International, 1997.

FURLAN, J. D. **Modelagem de objetos através da UML**. São Paulo: Makron Books, 1998.

GANE, C. **Análise estruturada de sistemas**. Rio de Janeiro: LTC, 1983.

GILLEANES, T. A. G. **UML**: uma abordagem prática. São Paulo: Novatec, 2004.

______. **UML**: uma abordagem prática. São Paulo: Novatec, 2006.

GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. São Paulo: Novatec, 2009.

IBM RATIONAL UNIFIED PROCESS. Versão 2003.06.12.01 (2005). Disponível em: <http://www-306.ibm.com/software/awdtools/rup/>. Acesso em: 13 set. 2005.

IBM RUP (2004) Disponível em: <http://www-130.ibm.com/developerworks/rational/products/rup>. Acesso em: 12 set. 2005.

IEEE COMPUTER SOCIETY. **IEEE Recommended Practice for Software Requirements Specifications**. Nova Iorque, EUA: The Institute of Electrical and Electronics Engineers, 1993.

KROLL, P.; KRUCHTEN P. **The Rational Unified Process Made Easy**: a Practitioner's Guide to the RUP, Addison Wesley, 2003.

LARMAN, Craig. **Applying UML and Patterns**: an introduction to object-oriented analysis and design and the unified process. Prentice Hall, 2001.

LIESENBERG, H. **Diagramas de colaboração**. Disponível em: <http://www.unicamp.br/~hans/mc426/#program>. Acesso em: 10 ago. 2005.

LIMA, A. S. **UML 2.0**: do requisito à solução. São Paulo: Erica, 2005.

MACHADO, F. M.; MAIA, L. P. **Arquitetura de sistemas operacionais**. 3. ed. Rio de Janeiro: LTC, 2002.

MASTALARI, N. **Contribuição ao processo de integração de informações da manufatura para empresas de pequeno e médio porte**. 2004. Tese (Doutorado em Engenharia Mecânica) - Faculdade de Engenharia Mecânica - Universidade Estadual de Campinas. Campinas, SP, 2004.

MAZOLLA, V. **Engenharia de software**. Disponível em: <http://pt.scribd.com/doc/48071734/Vitorio-Bruno-Mazzola-Engenharia-de-Software>. Acesso em: 24 abr. 2011.

NIEDERAUER, Mastelari. **Contribuição ao processo de integração de informações da manufatura para empresas de pequeno e**

médio porte. 2004. Dissertação (Mestrado em Engenharia Mecânica) - Universidade Estadual de Campinas / Faculdade Engenharia Mecânica, Campinas, SP, 2004.

PACHECO, R.; MONTENEGRO, F. **Orientação a objetos em C++**. São Paulo: Ciência Moderna, 1994.

PAGES-JONES, M. **Fundamentos do desenho orientado a objetos**. São Paulo: Makron Books, 2001.

PAULA FILHO, Wilson de Pádua. **Engenharia de software**: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2001.

PETERS, J. F.; PEDRYCZ, W. **Engenharia de software**: teoria e prática. Rio de Janeiro: Campus, 2001.

PRESSMAN, Roger. **Engenharia de software**. São Paulo: McGraw-Hill, 2002.

ROSENBERG, D.; STEPHENS, M.; COLLINS-COPE, M. **Agile development with ICONIX process:** people, process, and pragmatism. [S.L.]: Apress L. P., 2005.

RUMBAUGH, J. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Campus, 1994.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with SCRUM**. Prentice-Hall, 2002.

SILVA, A. M. R.; VIDEIRA C. A. E. **UML**: metodologias e ferramentas case. Lisboa: Centro Atlântico, 2001.

SOARES, M. S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **RESI - Revista Eletrônica de Sistemas de Informação**. 4 ed. Ano III, vol. III, n. I. nov. 2004.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. São Paulo: Addison-Wesley, 2003.

SOUZA, Francisco Flavio de; BRAGA, Rosana Teresinha Vaccare. Um método de desenvolvimento de sistemas de grande porte baseado no processo RUP. In: 1º Simpósio Brasileiro de Sistemas de Informação. **Anais do 1º SBSI**. Porto Alegre, 2004. p. 31-38.

SUTHERLAND, Jeff. **Agile development**: lessons learned from the first scrum. Cutter Agile Project Management Advisory Service: Executive Update, 2004.

TONSIG, Sérgio Luiz. **Engenharia de software:** análise e projeto de sistemas. São Paulo: Futura, 2003.

YOURDON, Edward. **Análise estruturada moderna**. Rio de Janeiro: Campus, 1992.

WUESTEFELD, Klaus. **Xispê: Extreme Programming**. 2001. Disponível em: <http://www.xispe.com.br/index.html>. Acesso em: 18 mar. 2002.

Sobre a professora conteudista

Vera Rejane Niedersberg Schuhmacher é mestre em Engenharia de Produção com ênfase em Usabilidade de Software pela Universidade Federal de Santa Catarina (UFSC). Professora da Unisul desde 1998, em disciplinas oferecidas nos cursos de Ciência da Computação e Sistemas de Informação e Pós-graduação. Pesquisadora do Núcleo de Computação, atua como coordenadora do Núcleo de Pesquisas em Usabilidade (NPU) prestando consultoria em Engenharia de *Software* e Usabilidade em empresas de tecnologias e projetos financiados por órgãos de fomento como Finep, CNPq e Funcitec.

Respostas e comentários das atividades de autoavaliação

A seguir são apresentadas respostas curtas e comentários sobre as atividades de autoavaliação propostas durante as unidades. Para melhor aproveitamento de seus estudos, realize a sua conferência somente depois de fazer as atividades.

Unidade 1

1) a) V

b) V

c) V

d) F

e) V

2) Sequência correta: G, D, C, B, E, F, A, D.

3) A afirmativa correta é (b).

4) a) E

b) P

c) I

d) C

5) A alternativa correta é: (a).

6) A alternativa correta é: (c).

O número de funcionalidades é bastante pequeno, por outro lado as regras de negócio sugerem uma certa confusão para o bom entendimento do analista, o que torna apropriado o uso do modelo prototipação para a identificação de requisitos.

7) Observe o modelo XP e o modelo SCRUM, e a seguir descreva o que é possível determinar como diferenças fundamentais em relação aos modelos tradicionais.

São citadas abaixo características existentes em ambos os modelos:

Tanto o Scrum como o XP apontam como de grande importância a comunicação entre a equipe e seus colaboradores; isso se evidencia por meio de reuniões formais e informais.

Outro aspecto bastante diferenciado está relacionado ao volume de documentação dos projetos; em ambos os modelos procura-se racionalizar evitando o excesso.

Um terceiro aspecto é a participação do usuário de forma efetiva no processo de desenvolvimento.

Os modelos são apontados como ideais para equipes pequenas e com interações rápidas.

Unidade 2

1) A alternativa correta é: (b).

2) A sequência correta é: d, a, c, b.

3) Relatório de análise do problema.

Observe que foram acrescentadas informações com o intuito de mostrar os itens de forma mais completa.

1 – Nome da Empresa: Clínica Bem-Estar

2 – Contato: Sr. Julibio Ritz (gerente) – Fone : 3339090
Cel.: 9987878

3 – Descrição do problema.

A clínica possui 34 médicos cadastrados em diferentes especialidades e presta atendimento a pacientes conveniados aos planos Bruxtr, Vpfzm e UIOlk ou particular.

A clínica funciona com um pequeno número de atendentes responsáveis pela marcação de consultas, preenchimento inicial de dados cadastrais. Cada médico faz 3 plantões semanais de 4 horas seguidas, as consultas possuem um intervalo de 30 minutos. Existe a possibilidade de a consulta ser de retorno, neste caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde são marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente que contém nome, endereço, telefone, data de nascimento, convênio.

O médico, ao atender o paciente, preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

4 – Identificação do principal objetivo do cliente.

O objetivo principal é o aumento na eficiência e tempo de resposta no processo de marcação de consultas.

5 – Descrição dos usuários do sistema.

Médicos – Todos os 34 médicos possuem especialização, todos possuem conhecimentos básicos de informática e noção do funcionamento e procedimentos da clínica. A faixa etária se encontra entre 30 e 48 anos. Nenhum médico possui qualquer deficiência física.

Atendente – As 2 atendentes possuem o segundo grau completo, razoável experiência em informática e conhecimento do processo de funcionamento da clínica. As atendentes estão na clínica há aproximadamente 3 anos.

Paciente – Existem aproximadamente 800 fichas na clínica. A clínica não sabe precisar se sua clientela possui acesso à internet. A clínica possui aproximadamente 85% das consultas realizadas por convênio.

6 – Descrição detalhada dos processos existentes (COMO O SISTEMA ATUAL FUNCIONA?).

Marcação da consulta – O paciente pode realizar o agendamento da consulta pessoalmente ou por telefone; em qualquer dos dois métodos os procedimentos são idênticos.

O paciente solicita a consulta informando o nome do médico ou a especialidade desejada, posteriormente informa a data desejada. A atendente verifica a possibilidade de marcação da consulta (observando se o médico em questão possui horário vago para a data desejada). Se existe horário disponível, a atendente solicita ao paciente o tipo de convênio ou se é particular. Se for convênio, é verificado se é um convênio válido; se for particular, é informado o valor da consulta. A atendente atualiza a agenda com o nome do paciente e o tipo de consulta (convênio/particular). O tempo para cada consulta é de 20 minutos ou 15 minutos para retorno. O médico possui intervalo de 10 minutos.

A consulta pode ser uma consulta de retorno, nesse caso a atendente verifica se a data está ainda dentro do prazo de retorno de 15 dias. Em caso afirmativo, a consulta é marcada na agenda.

Médico indisponível – Caso o médico solicitado esteja indisponível, a atendente procura informar o nome de outro médico disponível naquele horário ou o próximo horário disponível.

Atendimento – Se o paciente já possui cadastro, o mesmo é convidado a adentrar no consultório do médico. A partir desse momento, o médico solicita informações procedimentais para o futuro diagnóstico, preenchendo a ficha do paciente.

Finalizada a consulta, o paciente é liberado e a ficha é recolhida pela atendente, sendo novamente arquivada.

Se o paciente for novo, a atendente solicita o preenchimento da ficha cadastral com dados pessoais.

7 – Itens produzidos no sistema (quais são os relatórios e consultas existentes ou solicitados pelos clientes).

Nesse momento a clínica possui os seguintes documentos:

Agenda – São anotados o nome do paciente (60 caracteres), horário (hora/minuto) e convênio (Bruxtr, Vpfzm e UIOlk ou particular).

Ficha do paciente – Nome (60 caracteres), endereço (60 caracteres), telefone (12 caracteres), data de nascimento (date), convênio (Bruxtr, Vpfzm e UIOlk ou particular).

Ficha médica – Apresenta peso do paciente (numérico (03V2), altura (numérico (2V2), idade (numérico 3), motivo da consulta (Alfanumérico 100), queixa principal (Alfanumérico 100), doenças anteriores (Alfanumérico 150), diagnóstico (Alfanumérico 150), prescrição (Alfanumérico 200). A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

Relação de horários médicos – A listagem contém nome do médico (Alfanumérico 50), data (date) e horário (numérico 2V2) de atendimento do médico na clínica.

Relação de atendimentos por tipo – O relatório é emitido mensalmente, apresentando o nome do convênio, listando a partir dele o nome do médico e o total de consultas realizadas para o convênio. Ao final é apresentado o total de atendimentos por convênio.

Obs.: 3V2 significa uma variação numérica de até 999,99.

8 – Volume de informações do sistema atual.

A clínica possui aproximadamente 800 pacientes cadastrados, 34 médicos ativos e 2 atendentes. Apresenta convênio médico com 3 empresas, mas possibilidades de aumentar esse número. A clínica realiza em torno de 56 consultas por dia.

9 – Descrição de situações consideradas críticas e atores envolvidos.

A clínica apresenta situações críticas relacionadas à marcação incorreta de horário, com médico indesejado ou mesmo data e horário indesejados. As atendentes poderiam ser orientadas para telefonar ao paciente confirmando a consulta com três horas de antecedência.

10 – Restrições do projeto.

O cliente não deseja dispender recursos com a plataforma de sistema operacional e o banco de dados, sendo que deve ser considerada uma possibilidade *open source*.

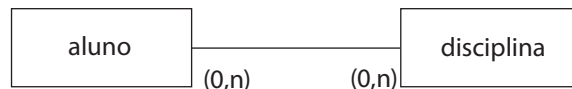
Unidade 3

1) Sequência correta é: B, G, D, F, C, E, D.

2) a) Um professor leciona várias disciplinas em sua universidade.

b) A universidade emprega vários funcionários.

c) Os funcionários são lotados em um departamento.

d) Um aluno pode estar matriculado em nenhuma ou várias disciplinas.

Unidade 4

1) As alternativas corretas são: (a) e (b).

2) A sequência é:

a) C

b) O

c) O

d) O

e) C

3) A sequência correta é:

a) Poliformismo

b) Encapsulamento

c) Mensagem

d) Herança

4) A sequência correta é:

a) A

b) B

c) D

d) C

e) A

f) E

g) B

Unidade 5

1) A afirmativa correta é: (b).

2) A sequência correta é:

a) V

b) F

c) F

d) V

e) F

f) V

3) A sequência correta é:

a) A

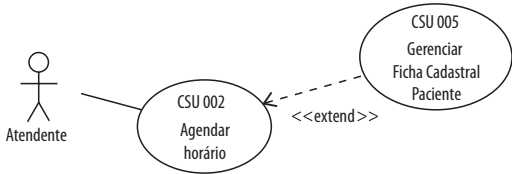
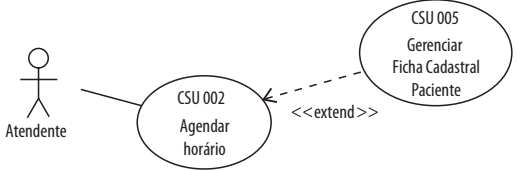
b) B

c) C

d) D

e) B

f) D

g) A

4) A definição dos requisitos funcionais:

RF01	O sistema deve permitir o gerenciamento de funcionários e seus dados cadastrais
RF02	O sistema deve controlar o sistema de acesso de acordo com as permissões de cada ator.
RF03	Deve ser possível realizar o gerenciamento de horário dos funcionários.
RF04	O sistema deve possibilitar o lançamento de horários de consulta por meio de uma agenda médica.
RF05	Deve ser possível a consulta de horários marcados por médico, por data.
RF06	O sistema deve possibilitar o gerenciamento de paciente (cadastro e ficha médica).
RF07	Deve ser possível incluir novos convênios ou mesmo excluir convênios com os quais a clínica opera.
RF08	É necessário que o sistema ofereça relatórios estatísticos de atendimento por convênio.
RF09	É necessário que o sistema emita relatórios estatísticos de atendimento por área de especialização.

A definição dos atores:

Ator	Descrição	Responsabilidade
Médicos	Todos os 34 médicos possuem especialização, todos possuem conhecimentos básicos de informática e noção do funcionamento e procedimentos da clínica. A faixa etária se encontra entre 30 e 48 anos. Nenhum médico possui qualquer deficiência física.	Gerenciamento de paciente.
Atendente	As 2 atendentes possuem o segundo grau completo, razoável experiência em informática e conhecimento do processo de funcionamento da clínica. As atendentes estão na clínica há aproximadamente 3 anos.	Gerenciamento de horário dos funcionários. Gerenciamento de agenda médica. Permitir consulta de horários marcados por médico, por data. Gerenciamento de paciente (cadastro).
Paciente	Pacientes com faixa etária até 14 anos, sob custódiaa e agendamento dos pais.	Gerenciamento de agenda médica (marcar consulta).
Gerente	Médico especialista possui conhecimentos básicos de informática e noção do funcionamento e procedimentos da clínica. Possui 42 anos não apresenta nenhuma deficiência física.	Acesso a todas as funcionalidades.

Os diagramas de 3 casos de uso e a tabela de documentação do caso de uso:

Caso de uso: CSU001 Gerenciamento de Funcionário	
Breve descrição	Gerenciamento do cadastro de dados pessoais e horários de trabalho de funcionários da clínica.
Ator primário	Gerente
Ator secundário	Funcionário
Precondições	O gerente estar devidamente logado no sistema.
Fluxo principal	1. O gerente solicita uma inclusão de funcionário; 2. O gerente informa o nome do funcionário; 3. O sistema informa o código do funcionário; 4. O gerente informa os dados pessoais do funcionário; 5. O gerente informa os horários em que o mesmo estará na clínica(CSU); 6. O registro do funcionário é armazenado.
Fluxo alternativo e exceções	No item 2, caso o nome já exista. Nesse caso: 1.1 São apresentados os dados cadastrais do funcionário; 1.2 São apresentadas as possibilidades de alterar, excluir ou finalizar.
Pós-condições	Dados do funcionário armazenados.
Requisito funcional	RF01– O sistema deve permitir o gerenciamento de funcionários e seus dados cadastrais.
Regras de negócio	RN01– O funcionário do tipo médico só pode ser excluído se não houver nenhuma consulta agendada.

Caso de uso: CSU002 Agendamento de Horário	
CSU 005 Gerenciar Ficha Cadastral Paciente Atendente CSU 002 Agendar horário <<extend>>	
Breve descrição	O caso de uso permite o agendamento do horário de consulta do paciente.
Ator primário	Atendente
Ator secundário	Paciente
Precondições	- O atendente estar devidamente logado no sistema; - horários médicos disponibilizados.
Fluxo principal	1. O paciente informa o nome do médico, data e horário desejado; 2. O atendente solicita a consulta para o médico informado nas datas solicitadas; 3. O atendente verifica se é primeira consulta na clínica. a. Se sim, executa o caso de uso gerenciar ficha cadastral paciente 4. O atendente solicita o tipo de consulta; 5. O agendamento do horário é armazenado. 6. A atendente informa novamente data, hora e médico da consulta para o paciente.
Fluxo alternativo e exceções	No item 1, caso o paciente não informe o nome do médico. Nesse caso: a atendente sugere o nome de um dos médicos segundo sua especialidade. No item 2, caso não haja horário disponível para o médico. Nesse caso: a atendente sugere outro horário, ou o nome de um segundo médico segundo sua especialidade.
Pós-condições	Consulta marcada.
Requisito funcional	RF04 – O sistema deve possibilitar o lançamento de horários de consulta por meio de uma agenda médica.
Regras de negócio	RN02– se o tipo de consulta for retorno, agendar somente 10 minutos na agenda. Se for consulta normal, agendar 20 minutos.

Caso de uso: CSU003 Cadastro de Convênios	
Gerente — CSU00 3 - Gerenciamento de Convênios	
Breve descrição	Neste caso ocorre o gerenciamento dos convênios.
Ator primário	Gerente
Ator secundário	
Precondições	O gerente deve estar devidamente logado no sistema.
Fluxo principal	1. O gerente solicita uma inclusão de convênio., 2. Informado o nome do convênio. 3. O sistema informa o código interno do convênio. 4. São informados dados cadastrais do convênio. 5. Os dados são armazenados.
Fluxo alternativo e exceções	No item 2, caso o nome já exista. Nesse caso: são apresentados os dados cadastrais do funcionário; são apresentadas as possibilidades de alterar, excluir ou finalizar.
Pós-condições	Consulta marcada.
Requisito funcional	RF07– Deve ser possível incluir novos convênios ou mesmo excluir convênios com os quais a clínica opera.
Regras de negócio	

Unidade 6

1) As alternativas corretas são: (**b)** e (**d)**.

2) A sequência é:

a) c

b) a

c) a

d) b

e) c

f) b

3) A sequência é:

a) G

b) B

c) C

d) E

e) F

f) D

g) A

4)

a) As classes persistentes.

É possível identificar:

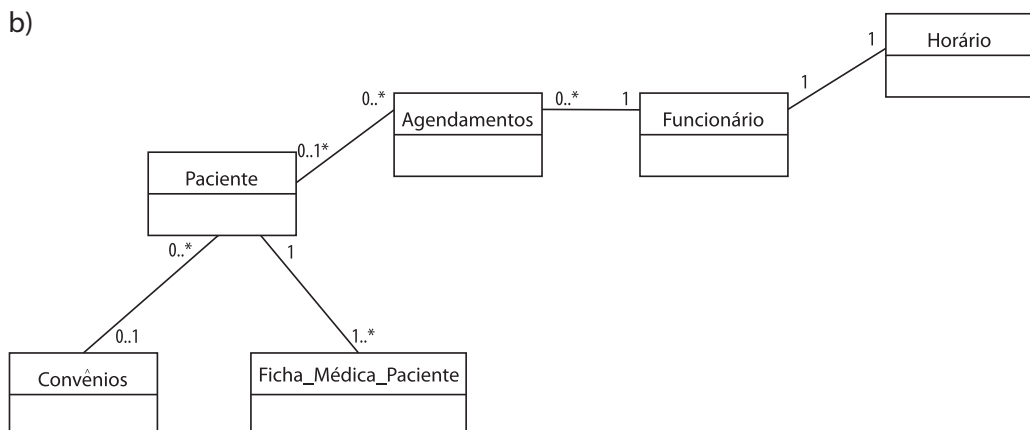
A classe Paciente – que armazena os dados cadastrais do paciente.

A classe Agendamentos – que armazena o horário das consultas, nome do paciente e médico.

A classe Funcionário – armazena os dados dos funcionários, inclusive do funcionário médico.

A classe Horário – que armazena o horário de atendimentos da equipe médica

A classe Ficha Médica – armazena a ficha de atendimento do paciente.

b)

Unidade 7

1) A sequência correta é:

a) B

b) D

c) F

d) A

e) H

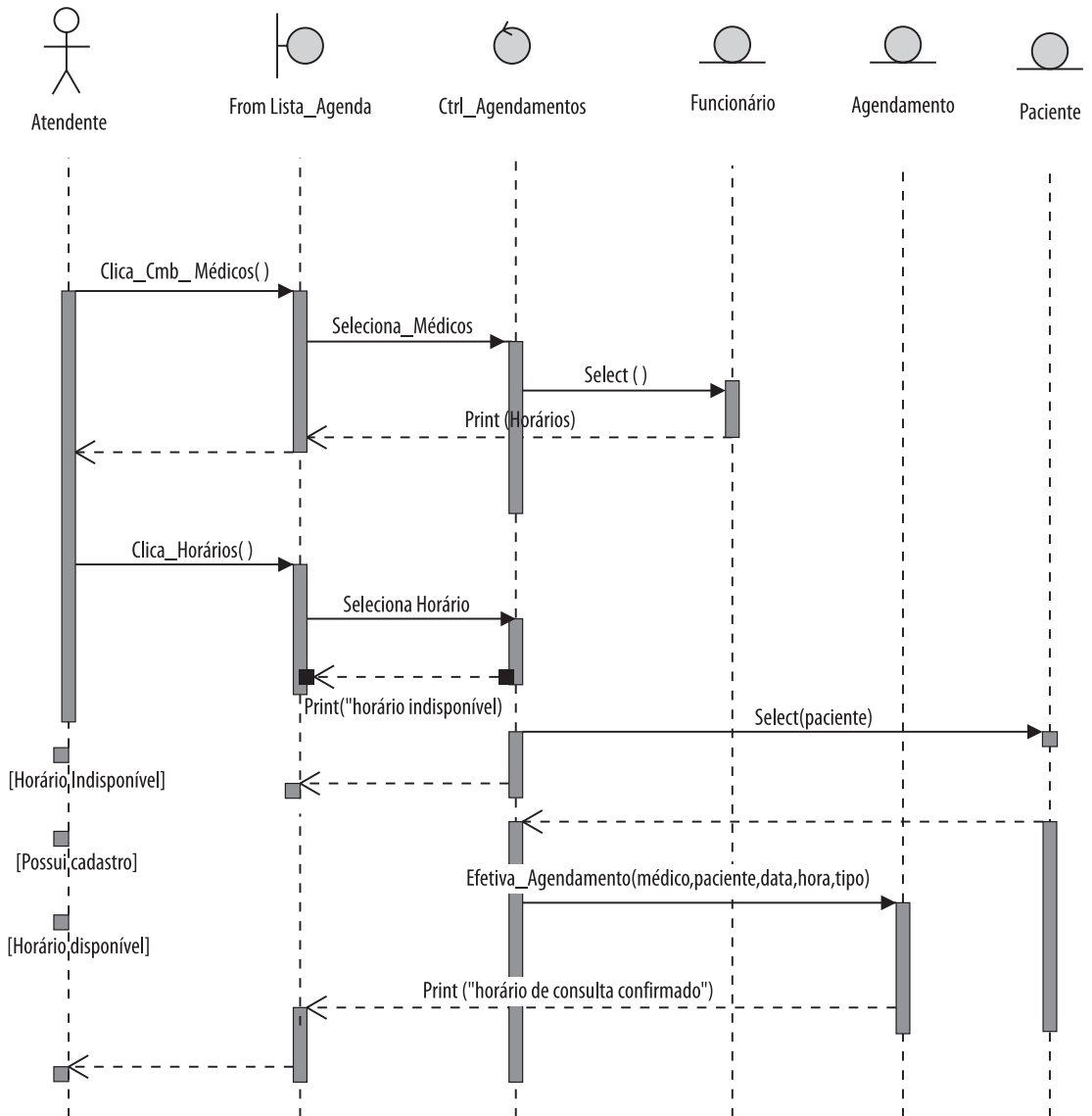
f) E

g) G

h) I

i) C

2) No diagrama a seguir, são descritos dois diagramas: o Listar Agenda e o Agendamento do Horário.

CSU002– Agendamento de Horário

3) Alternativa correta: b.

Unidade 8

1) a) Esta modificação é chamada de transição entre estados.

b) Os estados representam o resultado de atividades executadas por um objeto.

c) O estado final indica o final do ciclo de vida de um objeto.

d) Os objetos de um sistema modificam seu estado de forma análoga a objetos do mundo real.

2) a) DTE

b) DA

c) DTE

d) DA

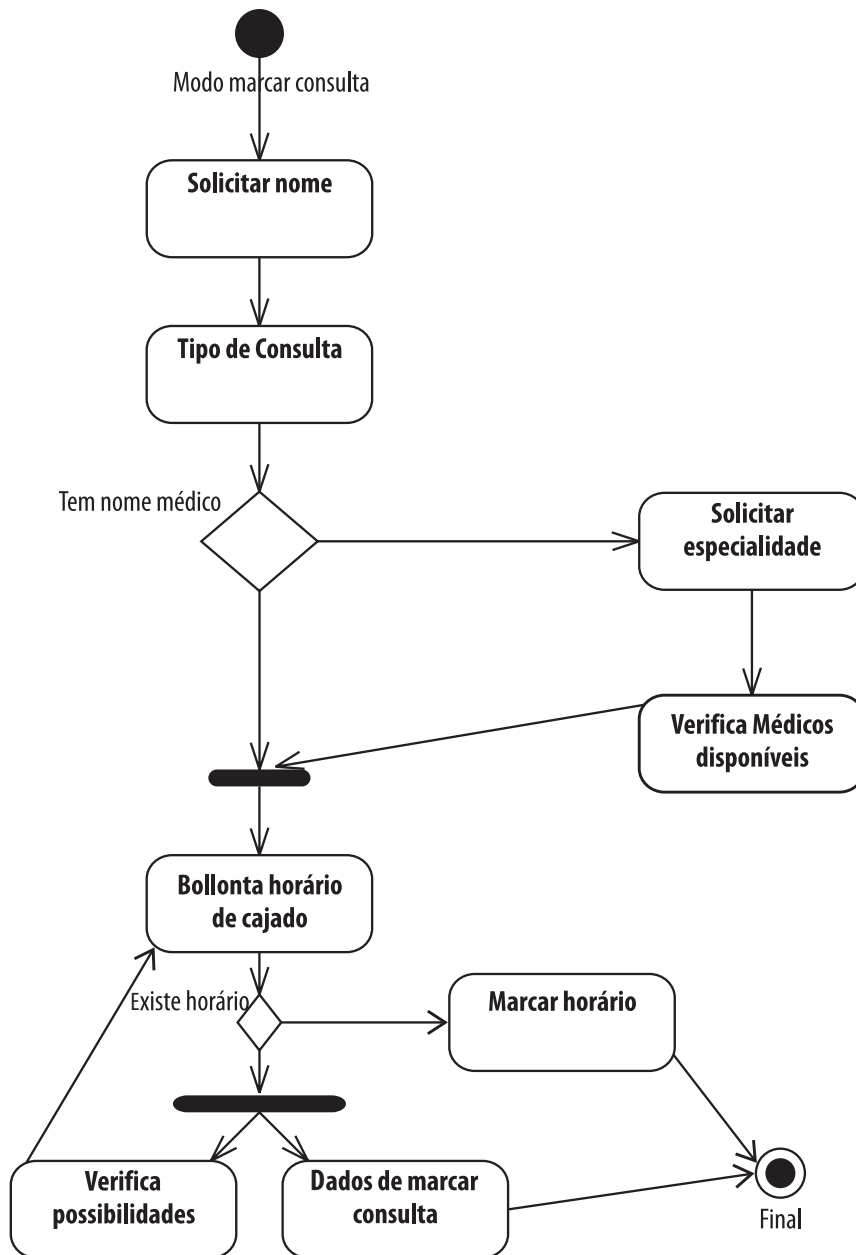
3) a) E

b) B

c) A

d) C

e) E

4) Diagrama de atividades do caso de uso Marcar Consulta da Clínica Bem-Estar a partir da visão do ator Atendente.

Unidade 9

1) As afirmativas corretas são: (**b)** e (**d)**.

2) Um papel é uma definição abstrata de um conjunto de atividades executadas e dos respectivos artefatos.

Analista de Sistemas

O papel do Analista de Sistemas é liderar e coordenar a identificação de requisitos e a modelagem de casos de uso, delimitando o sistema e definindo sua funcionalidade; por exemplo, estabelecendo quais são os atores e casos de uso existentes e como eles interagem.

Analista de Teste

O papel do Analista de Teste é inicialmente identificar e posteriormente definir os testes necessários, monitorar a abrangência dos testes e avaliar a qualidade geral obtida ao testar os Itens de Teste-alvo. Este papel também envolve a especificação dos Dados de Teste necessários e a avaliação do resultado dos testes conduzidos em cada ciclo de teste

3) a) E

b) A

c) C

d) D

e) B

4) O modelo utilizado será dependente do tipo de produto a ser desenvolvido e da dinâmica da empresa desenvolvedora, mas podemos citar 3 modelos que podem ser considerados interessantes em qualquer tipo de projeto:

- Modelo de domínio que procura estabelecer o contexto do sistema;
- Modelo de caso de uso que estabelece os requisitos funcionais do sistema;
- Modelo do processo que estabelece os mecanismos de concorrência e de sincronização do sistema.

Biblioteca Virtual

Veja a seguir os serviços oferecidos pela Biblioteca Virtual aos alunos a distância:

- Pesquisa a publicações on-line
  <www.unisul.br/textocompleto>
- Acesso a bases de dados assinadas
  <www.unisul.br/bdassinadas>
- Acesso a bases de dados gratuitas selecionadas
  <www.unisul.br/bdgratuitas >
- Acesso a jornais e revistas on-line
  <www.unisul.br/periodicos>
- Empréstimo de livros
  <www.unisul.br/emprestimos>
- Escaneamento de parte de obra*

Acesse a página da Biblioteca Virtual da Unisul, disponível no EVA, e explore seus recursos digitais.

Qualquer dúvida escreva para: bv@unisul.br

---

* Se você optar por escaneamento de parte do livro, será lhe enviado o sumário da obra para que você possa escolher quais capítulos deseja solicitar a reprodução. Lembrando que para não ferir a Lei dos direitos autorais (Lei 9610/98) pode-se reproduzir até 10% do total de páginas do livro.